ORSTOM

Institut Français de Recherche Scientifique pour le Développement en Coopération

---

Notice OVNIh du laboratoire d'hydrologie # 1

SIMPLE et ROSEN :
Deux méthodes d'optimisation non linéaire

Théorie et pratique

Eric Servat
Alain Dezetter

Juin 1988

Les notices **OVNIh*** du Laboratoire d'Hydrologie de l'ORSTOM

L'**Organisation** du traitement des données hydrologiques passe de plus en plus par le développement de logiciels utilisant les méthodes de l'analyse numérique et statistique en les adaptant au cas particulier de la **Valorisation** des connaissances hydrologiques. La mise au point de notices **Normalisées** a pour but de faciliter la documentation de ces logiciels de traitement de l'**Information hydrologique** et de leur assurer ainsi une meilleure diffusion en rendant leur emploi plus accessible à l'utilisateur non spécialisé. Les notices OVNIh sont destinées avant tout aux hydrologues de l'ORSTOM ou d'Instituts partenaires dont les besoins en la matière sont connus et qui du fait de leur dispersion géographique sont souvent privés d'un accès facile à ce type d'outils sur leur lieu de travail. Cette priorité n'exclut naturellement pas la diffusion auprès d'autres hydrologues francophones qui seraient éventuellement intéressés par l'utilisation de ces logiciels ou la consultation de leur documentation.

Le Laboratoire d'Hydrologie joue uniquement un rôle d'animation en veillant à ce que les notices publiées dans cette série soient normalisées dans leur présentation et dans leur mode de description des fonctions du logiciel.

Les auteurs (qu'ils soient les rédacteurs de la documentation ou les concepteurs des logiciels) gardent la totale maîtrise et responsabilité de leur produit. La diffusion de notices normalisées doit leur permettre d'accroître le bénéfice qu'ils retireront de leurs travaux. Les améliorations éventuelles et la maintenance sont du ressort des auteurs, interlocuteurs des utilisateurs dans ce domaine.

En avant-propos de chaque notice figurent les noms des gens ayant pris part au développement du logiciel et à la rédaction de la notice, le cadre dans lequel ils ont effectué leur travail (unité de recherche ORSTOM, Programmes, Instituts ou Organismes en cas de collaboration avec des personnes n'appartenant pas à l'ORSTOM), et l'appréciation qu'ils portent eux-mêmes sur leur produit.

Les notices OVNIh ne sont donc pas une nouvelle collection des éditions de l'ORSTOM, elles ne sont pas commercialisables, elles sont librement reproductibles et le Laboratoire d'Hydrologie se réserve simplement le droit de demander une participation financière aux coûts de reproduction des exemplaires qu'il diffusera. En règle générale des disquettes contenant les codes exécutables seront données à quiconque en fera la demande, à charge pour celui-ci de fournir les disquettes vierges.

N.B : La mise en oeuvre de cette série de notice est une initiative de l'équipe de chercheurs travaillant au sein du programme U.L.M de l'U.R 604 de l'ORSTOM qui a décidé d'en placer la réalisation sous la responsabilité du laboratoire d'hydrologie pour des raisons opérationnelles évidentes. Toute suggestion ou contribution peut donc être adressée au responsable du programme ou a celui du laboratoire d'hydrologie.

---

* **Organisation, Valorisation** et **Normalisation** de l'**Information hydrologique.**

AVANT-PROPOS

SIMPLE et ROSEN sont deux sous-programmes écrits en **FORTRAN 77**, pouvant être interfacés avec tout programme qui requiert l'optimisation d'une fonction critère.

Les deux méthodes présentées ici sont des méthodes numériques développées d'une part par Nelder et Mead et d'autre part par Rosenbrock.

La mise en oeuvre de ces sous-programmes est immédiate; en particulier ils ne font appel à aucune autre ressource extérieure que la bibliothèque FORTRAN accompagnant tous les compilateurs à la norme 77.

Cette notice doit être considérée a la fois comme une présentation rapide des algorithmes d'optimisation et comme un guide pour l'utilisateur des routines **SIMPLE et ROSEN**.

Ce travail a été réalisé au centre ORSTOM de Montpellier puis d'Adiopodoumé (Côte d'Ivoire) dans le cadre du programme de recherche **Utilité et Limites des Modeles en hydrlologie (U.L.M.)**.

SOMMAIRE

1. PRESENTATION GENERALE

Les puissants moyens de calcul offerts par l'informatique sont à la base de l'intérêt suscité actuellement par les méthodes de programmation mathématique. Parmi celles-ci, les techniques d'optimisation, fréquemment développées dans le cadre de la programmation non linéaire, occupent une place importante.

Il n'y a généralement pas de réponse unique aux questions que se posent les scientifiques. Dans ces conditions, pour un problème donné, procéder à une optimisation a pour objectif de déterminer quelle est la meilleure des solutions potentielles. Associé à ces techniques, un critère numérique est très souvent utilisé pour évaluer les solutions possibles et retenir la meilleure d'entre elles (au sens de ce critère).

On peut optimiser de différentes manières qui vont de l'utilisation de la simple arithmétique à celle de procédures numériques et analytiques particulièrement sophistiquées.

Les méthodes d'optimisation peuvent être classées en plusieurs catégories:

+) *les méthodes analytiques*, qui utilisent les techniques classiques du calcul différentiel. Elles sont généralement peu satisfaisantes dés lors que l'on a à traiter des problèmes fortement non linéaires. (Méthode du Maximum de vraisemblance par exemple)

+) *les méthodes numériques*, qui emploient des procédures itératives. On peut les utiliser pour résoudre des problèmes vis à vis desquels les méthodes analytiques sont inefficaces, notamment en cas de forte non linéarité. Les techniques d'optimisation que nous présenterons ici se rattachent à cette catégorie. (Méthodes de Rosenbrock, Nelder et Mead, Powell ou Marquardt par exemple)

+) *les méthodes graphiques*. Si elles ont l'avantage d'être simples et immédiates quant à leurs conclusions, elles ont le tort de ne pouvoir s'appliquer qu'à des fonctions d'une ou deux variables indépendantes au plus. (Méthode des déviations résiduelles par exemple)

+) *les méthodes expérimentales.* Dans ce cas, on tentera d'approcher l'extremum de la fonction critère en agissant directement sur les variables caractéristiques du problème étudié. Les résultats de l'essai n sont utilisés pour définir l'essai n+1. On procède ainsi de proche en proche jusqu'à un résultat jugé satisfaisant.

+) *les méthodes "étude de cas".* Elles correspondent à l'évaluation directe d'un nombre restreint de solutions jugées représentatives du problème étudié. Dans ce cas, cependant, la solution retenue sera vraisemblablement "sous-optimale", car fortement teintée de subjectivité.

Ces techniques d'optimisation des représentations mathématiques de processus naturels sont très puissantes,et peuvent parfois apparaître comme la solution aux problèmes que se pose le scientifique, concernant la détermination des paramètres des modèles qu'il a élaboré. Une certaine prudence reste néanmoins indispensable. On se heurte, en effet, fréquemment à deux types de difficultés ayant trait d'une part, à la formulation du modèle proprement dit et de la fonction critère utilisée et, d'autre part, aux techniques numériques elles-même. Examinons les brièvement et de manière non exhaustive:

a) modèle et fonction critère

a-1) La fonction critère utilisée peut être insensible aux modifications apportées aux valeurs des paramètres du modèle. Il sera alors impossible de localiser précisément un extremum.

a-2) Le critère à optimiser peut être ponctuellement indéfini à l'intérieur de l'intervalle de recherche de l'extremum. C'est, parfois, le cas des expressions ayant un dénominateur polynomial. Il faudra, alors, restreindre l'intervalle de recherche en ajoutant des contraintes au problème ou reformuler la fonction critère.

a-3) On peut se trouver confronté à des problèmes d'échelle lorsque les termes qui constituent la fonction critère ont des ordres de grandeur par trop différents. Dans ce cas, le critère restera insensible aux variations affectant le terme le plus faible.

a-4) Certaines difficultés peuvent également être le fait de l'existence d'inter-relations entre les paramètres du modèle. Il est alors nécessaire d'en reprendre la formulation.

b) les méthodes numériques

b-1) Dans le cas de fonctions non linéaires il peut exister plus d'un extremum. En conséquence, si le jeu de paramètres initialement choisi est trop éloigné de la solution optimale, la méthode peut s'arrêter au niveau d'un faux extremum sans avoir, pour autant, atteint le véritable optimum. En pratique, les résultats d'études antérieures ou un raisonnement physique peuvent aider à la détermination d'un jeu de paramètres initial optimal. En dernier ressort il est toujours possible de tester plusieurs jeux initiaux et de s'assurer qu'ils conduisent tous à un extremum présentant une même valeur de critère.

b-2) Les erreurs dûes aux approximations numériques réduisent l'efficacité de nombreux algorithmes en étant souvent à l'origine des problèmes rencontrés en terme de convergence et de stabilité. L'approximation des dérivées par des schémas aux différences est également, dans certains cas, responsable du mauvais fonctionnement des techniques d'optimisation.

Comme on le voit ces méthodes ne peuvent être appliquées à un problème donné sans une importante réflexion préalable. La puissance des algorithmes mis en jeu ne dispense en rien l'utilisateur d'une conception rigoureuse du modèle mathématique retenu et du choix des procédures numériques appropriées. Ces techniques d'optimisation, très utilisées dans de nombreuses disciplines scientifiques, sont d'un intérêt incontestable pour les hydrologues, très souvent confrontés à des problèmes non linéaires mettant en jeu un grand nombre de paramètres.

Nous présenterons ici deux méthodes de programmation non linéaire très couramment utilisées et dont l'emploi sur micro-ordinateur ne présente aucune difficulté. Il s'agit des algorithmes développés par Rosenbrock et par Nelder et Mead.

2. DEUX METHODES D'OPTIMISATION NE NECESSITANT PAS LE CALCUL DES DERIVEES

2.1 Introduction

Nous décrirons ici deux méthodes d'optimisation ne nécessitant pas le calcul des dérivées (la méthode de Nelder et Mead, d'une part, celle de Rosenbrock, d'autre part).Dans ce cas, on parle souvent de méthodes de recherche directe conduisant à la solution par les seules évaluations successives de la fonction critère.

En régle générale, les méthodes basées sur le calcul des dérivées convergent plus rapidement vers la solution que les méthodes de recherche directe lorsqu'il s'agit de résoudre un problème d'optimisation non linéaire. On se heurte cependant à deux difficultés principales lorsque l'on est amené à les utiliser:

a) dès que le nombre de variables considéré est un tant soit peu important, il devient extrêmement difficile d'écrire les dérivées du premier ou du deuxième ordre sous une forme analytique. Le recours éventuel à des schémas aux différences finies introduit généralement une erreur numérique qui ne plaide pas en faveur de telles substitutions, notamment au voisinage de l'extremum.

b) les techniques d'optimisation basées sur l'évaluation des dérivées premières et éventuellement secondes demandent de la part de l'utilisateur, un travail préparatoire beaucoup plus important que dans le cas des méthodes de recherche directe.

Du fait de ces difficultés, des algorithmes d'optimisation par recherche directe ont été élaborés qui, bien que plus lents dans le cas de problèmes simples, peuvent se révéler très satisfaisants, en pratique, pour l'utilisateur.

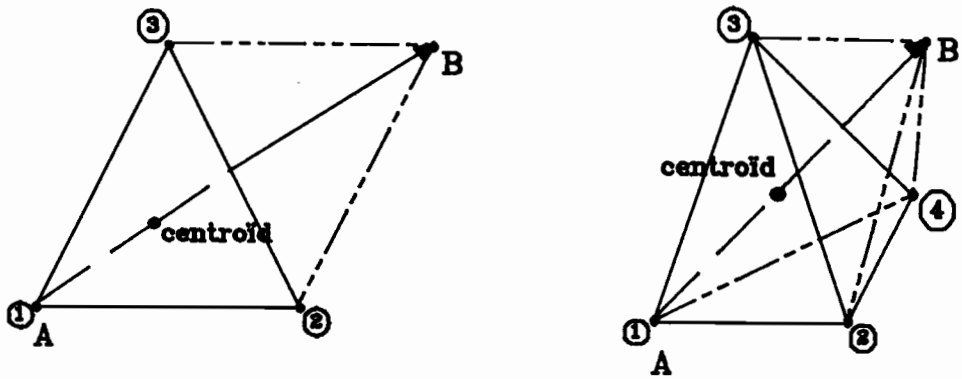
Nous présenterons ici deux de ces méthodes de recherche directe: la méthode élaborée par Nelder et Mead (1964), improprement appelée parfois "méthode du Simplex", et la méthode proposée par Rosenbrock (1960).

2.2 La méthode de Nelder et Mead

Nelder et Mead ont proposé une méthode qui s'est révélée être particulièrement efficace et simple à programmer.

Pour bien comprendre la technique qu'ils ont développée, nous décrirons tout d'abord rapidement un algorithme antérieur, la méthode du simplex de Spendley, Hext et Himsworth (1962).

Rappelons, avant tout, qu'un simplex est un polyèdre régulier dans un espace à n dimensions E^n. Par exemple, dans un espace à deux dimensions, un simplex régulier est un triangle équilatéral; dans un espace à trois dimensions un simplex est un tétraèdre régulier; etc. (cf figure 1)



Simplex à deux variables — Simplex à trois variables

Figure 1 Simplex pour deux et trois variables

Dans le cas de la recherche du minimum d'une fonction critère f(x),on peut sélectionner des jeux de valeurs de x en des points de E^n situés aux sommets du simplex. A partir de la géométrie analytique, on peut montrer que les coordonnées des sommets d'un simplex régulier composent une matrice D, dans laquelle les colonnes représentent les composantes des sommets (numérotés de 1 à n+1), et les lignes les coordonnées (de i=1 à n).

$$D= \begin{bmatrix} 0 & d_1 & d_2 \ldots\ldots & d_2 \\ 0 & d_2 & d_1 \ldots\ldots & d_2 \\ 0 & d_2 & d_2 \ldots\ldots & d_2 \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ 0 & d_2 & d_2 \ldots\ldots & d_1 \end{bmatrix} \quad \text{matrice n * (n+1)}$$

avec:

$$d_1= \frac{t}{n.\sqrt{2}} \left[\sqrt{(n+1)}+n-1 \right]$$

$$d_2= \frac{t}{n.\sqrt{2}} \left[\sqrt{(n+1)}-1 \right]$$

t= distance entre deux sommets

Par exemple, dans le cas où n=2 et t=1, le triangle de la figure 1 a les coordonnées suivantes:

	sommet		
	1	2	3
x_1	0	.965	.259
x_2	0	.259	.965

La fonction critère peut être évaluée en chacun des sommets du simplex. On peut alors faire une projection du point présentant la valeur la plus élevée passant par le barycentre des autres sommets (cas du point A dans la figure 1). Le point A est alors supprimé, et un nouveau simplex, obtenu par "réflexion", peut être constitué à partir des anciens points restants et du nouveau point projeté, B. L'utilisation de cette procédure, à savoir l'élimination systématique du sommet présentant la valeur la plus élevée de la fonction critère, et de quelques règles permettant de réduire la taille du simplex et de l'empêcher de se boucler sur lui même au voisinage de l'extremum, permettent une recherche directe à pas fixe mais à direction variable.

La figure 2 présente les simplex successifs formés dans un espace à 2 dimensions dans le cas d'une fonction critère simple.

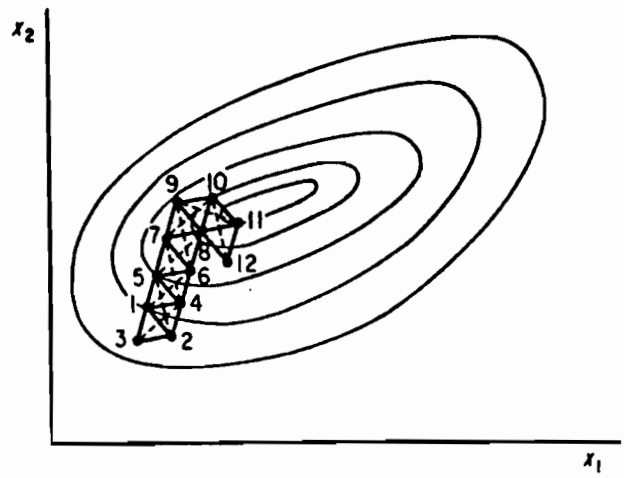


Figure 2: Séquence de simplexes réguliers obtenus en minimisant f(x)

Certaines difficultés pratiques, liées au fait qu'il n'y ait pas d'accélération possible de la recherche, et aux problèmes rencontrés par la méthode pour certaines formes de vallées ou de crêtes présentées par l'hypersurface correspondant à la fonction critère, ont conduit à concevoir différentes améliorations.

Dans le cas de la méthode de Nelder et Mead le simplex peut voir sa forme modifiée (en vertu de quoi il faudrait, d'ailleurs, ne plus parler de simplex). Si le principe général de la méthode est identique à celui que nous venons de voir, l'idée nouvelle est l'adaptation du simplex au problème local par accroissement ou contraction selon que l'on se rapproche ou non du minimum.

L'objectif est toujours de minimiser une fonction de n variables indépendantes en utilisant les (n+1) sommets d'un polyèdre d'un espace à n dimensions (E^n). Chaque sommet peut être défini par un vecteur x. Celui qui présente la plus forte valeur de f(x) est projeté selon une direction passant par le barycentre des sommets restants. L'amélioration (la diminution) de la fonction critère est obtenue en remplaçant ainsi successivement le point présentant la valeur la plus élevée de f(x) jusqu'à ce que le minimum soit atteint.

Les détails de l'algorithme sont les suivants (en se plaçant à une étape donnée k du processus de recherche, c'est à dire pour une structure de simplex donnée):

Soit $x_i = [\ x_{i1}, \ldots, x_{ij}, \ldots, x_{in}\ ]^T$ avec $i = 1, \ldots, n+1$ qui représente les coordonnées du sommet i dans l'espace E^n, et $f(x_i)$ la valeur de la fonction critère en x_i.

On peut, en outre, repérer les vecteurs associés aux sommets qui correspondent aux valeurs maximales et minimales de la fonction critère:

$$f(x_h) = \max [\ f(x_1), \ldots, f(x_{n+1})\ ]$$

avec la correspondance ad hoc: $x_h = x_i$

$$f(x_m) = \min [\ f(x_1), \ldots, f(x_{n+1})\ ]$$

avec la correspondance ad hoc: $x_m = x_i$

Puisque dans E^n le polyèdre est constitué de n+1 sommets auxquels sont associés les vecteurs $x_1, \ldots, x_{n+1}$, on conviendra d'appeler x_{n+2} le vecteur correspondant au barycentre des différents sommets à l'exception de x_h. Le point associé à x_{n+2} est appelé "centroid" par Himmelblau (1972). Ses coordonnées se calculent par:

$$x_{n+2,j} = (1/n) * [(\sum_{i=1}^{n+1} x_{ij}) - x_{hj}] \qquad j = 1,..,n \qquad \text{(eq. 1)}$$

l'indice j désignant chaque axe de coordonnées.

La procédure permettant de déterminer un sommet dans E^n pour lequel f(x) présente une meilleure valeur tient en quatre opérations:

a) *Réflexion*

On projette x_h par une direction passant par le "centroid" en calculant

$$x_{n+3} = x_{n+2} + \alpha(x_{n+2} - x_h) \qquad \text{(eq. 2)}$$

où $\alpha > 0$ est le coefficient de réflexion,
x_{n+2} le "centroid" calculé avec (eq. 1),
x_h le sommet correspondant à la valeur de f(x) la plus élevée.

b) *Extension*

Si $f(x_{n+3}) \leq f(x_m)$ on recalcule un nouveau point correspondant au vecteur x_{n+4} et permettant l'extension du simplex

$$x_{n+4} = x_{n+2} + \theta(x_{n+3} - x_{n+2}) \qquad \text{(eq. 3)}$$

où $\theta > 1$ est le coefficient d'extension.

Si $f(x_{n+4}) < f(x_m)$, on remplace le point correspondant à x_h par celui correspondant à x_{n+4} et on relance la procédure avec un calcul du nouveau "centroid". Dans le cas contraire, on remplace x_h par x_{n+3} et on procède de la même façon.

c) *Contraction*

Si $f(x_{n+3}) > f(x_i)$ pour tout i différent de h, on recalcule un nouveau point.

$$x_{n+5} = x_{n+2} + \beta(x_h - x_{n+2}) \qquad \text{(eq. 4)}$$

où $0 < \beta < 1$ est le coefficient de contraction.

On remplace alors le point correspondant à x_h par celui associé à x_{n+5} et on relance la procédure.

d) *Reduction*

Si $f(x_{n+3}) > f(x_h)$, on détermine un nouveau simplex en réduisant les coordonnées des vecteurs x_i, i= 1,..,n+1 par l'opération suivante:

$$x_i = x_m + \tfrac{1}{2}(x_i - x_m) \quad i=1,..,n+1 \qquad \text{(eq. 5)}$$

On relance alors la procédure en recalculant un nouveau "centroid".

Le critère d'arrêt de la recherche,utilisé par Nelder et Mead, a pour expression

$$\{ 1/(n+1) * \sum_{i=1}^{n+1} [f(x_i) - f(x_{n+2})]^2 \}^{\frac{1}{2}} \leq \epsilon \qquad \text{(eq. 6)}$$

dans laquelle ϵ est arbitrairement fixé alors que $f(x_{n+2})$ est la valeur de la fonction critère au "centroid".

L'algorithme proposé par Nelder et Mead est illustré par l'organigramme de la figure 3.

ORGANIGRAMME METHODE DE NELDER ET MEAD

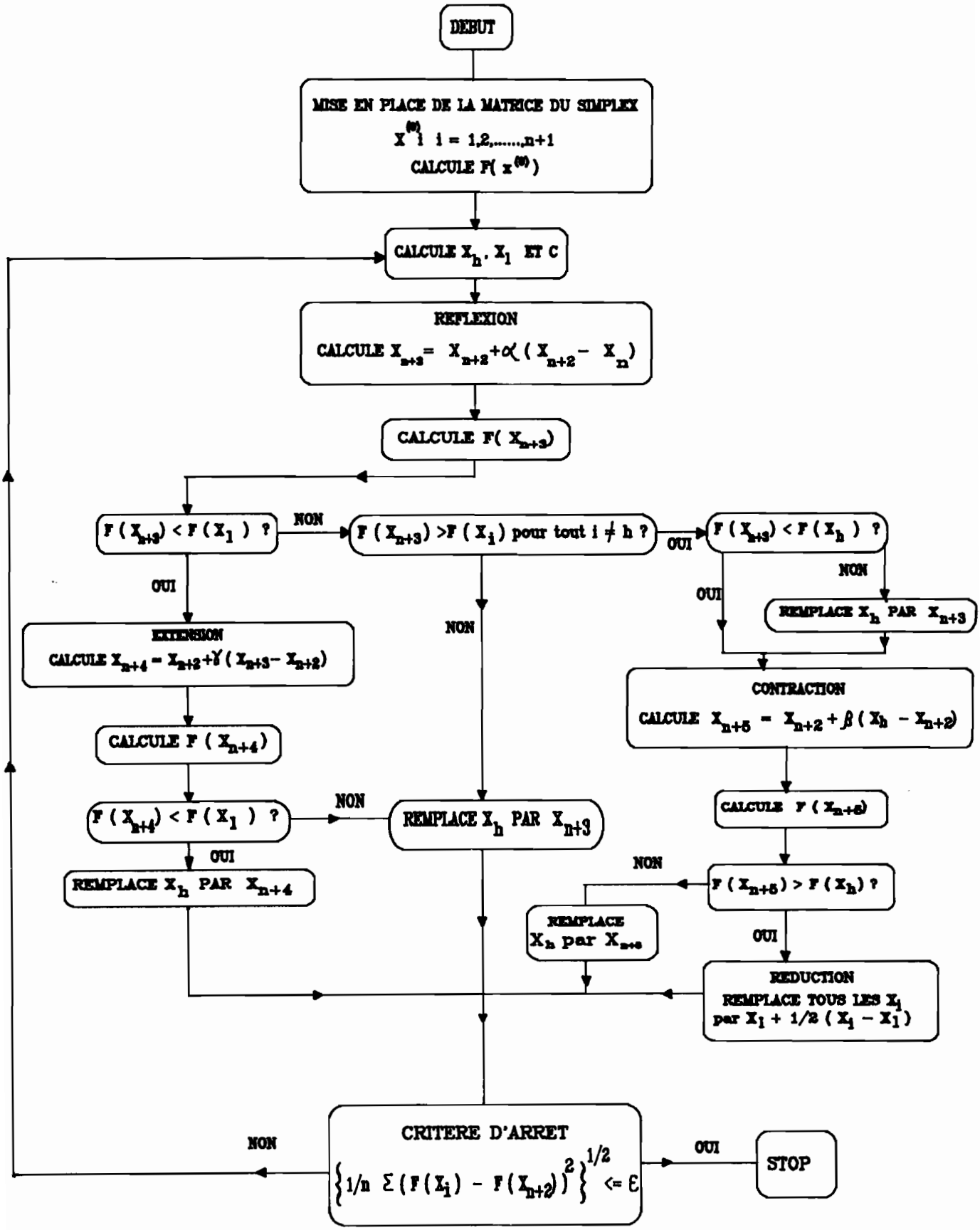


Figure 3

Les modifications que l'on peut apporter à la taille et à la forme du polyèdre lui permettent de s'adapter aux formes variées des hypersurfaces générées par les fonctions critères dans E^n.

Reste néanmoins le problème de la détermination des valeurs des paramètres α, β, et θ. Nelder et Mead d'une part, Paviani (1969) d'autre part, ont résolu un grand nombre de problèmes tests et sont arrivés aux conclusions suivantes:

- Pour Nelder et Mead les valeurs à utiliser sont:

$\alpha = 1$, $\beta = 0.5$, $\theta = 2$

- Paviani, quant à lui, recommande:

$\alpha = 1$
$0.4 \leq \beta \leq 0.6$
$2.8 \leq \theta \leq 3.0$

en remarquant que l'effet du choix de β est plus prononcé, en terme d'efficacité de la recherche, que celui de θ.

2.3 La méthode de Rosenbrock

La méthode élaborée par Rosenbrock (1960) est une procédure itérative basée, comme la technique précédente, sur la recherche de l'extremum d'une fonction critère préalablement définie. Partant d'un point donné, le point suivant, auquel est associé une meilleure valeur de la fonction critère, est déterminé par des recherches successives unidimensionnelles le long d'un jeu de directions orthonormées $s_1,\ldots,s_n$. Ces directions, générées par le processus d'orthogonalisation de Gram-Schmidt, correspondent aux axes d'un repère orthonormé d'un espace à n dimensions E^n.

Décrivons cette méthode plus en détails. Soit $s_1^k,\ldots, s_n^k$ des vecteurs unités de l'espace à n dimensions E^n, avec $k= 0,1,\ldots$ permettant d'identifier les étapes de la recherche. Les vecteurs orthonormés $s_1^k,\ldots, s_n^k$ sont générés à partir de l'information obtenue à l'étape (k-1) à l'aide des équations 7 et 8 détaillées plus loin.

Pour l'étape initiale, $k= 0$, les directions $s_1^0,\ldots, s_n^0$ sont habituellement choisies parallèles aux axes des $x_1,\ldots, x_n$.

En régle générale, les directions de recherche orthogonales peuvent être exprimées comme des combinaisons de toutes les coordonnées des variables indépendantes de la façon suivante:

$$s_1^k = a_{11}^k \delta_1 + a_{12}^k \delta_2 + \ldots + a_{1n}^k \delta_n$$
$$s_2^k = a_{21}^k \delta_1 + a_{22}^k \delta_2 + \ldots + a_{2n}^k \delta_n$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$s_n^k = a_{n1}^k \delta_1 + a_{n2}^k \delta_2 + \ldots + a_{nn}^k \delta_n$$

avec:
δ_i, vecteur unité dans la direction x_i
et a_{ij}, cosinus directeur de s_i.

A l'étape k de la recherche, posons $x_0^k = x_n^{k-1}$ comme étant le point de E^n auquel la procédure démarre. Définissons également $\emptyset_1, \ldots, \emptyset_n$ comme la longueur des pas associés respectivement aux directions $s_1, \ldots, s_n$.

Partant de x_0^k la recherche débute en introduisant une modification $\phi_1^k \, s_1^k$ dans la première direction de coordonnée. Si la valeur de la fonction critère $f(x_0^k + \phi_1^k \, s_1^k)$ est égale ou inférieure à $f(x_0^k)$ on a affaire à un succès et ce nouveau point remplace x_0^k, ϕ_1^k est alors multiplié par un facteur > 0 et c'est au tour de la direction de recherche s_2^k d'être modifiée. Si la valeur de $f(x_0^k + \phi_1^k \, s_1^k)$ est supérieure à $f(x_0^k)$, on a affaire à un échec, x_0^k n'est pas remplacé, $\emptyset_1^k$ est multiplié par un facteur $\beta < 0$ et c'est au tour de la direction de recherche s_2^k d'être modifiée. De manière générale, Rosenbrock recommande l'utilisation de $\alpha = 3$ et $\beta = -0.5$.

Après que chacune des n directions de recherche ait été modifiée, la première l'est à nouveau, d'un pas de longueur $\alpha \, \phi_1^k$ ou $\beta \, \phi_1^k$ selon le résultat précédent enregistré suivant cette direction s_1^k. On continue ainsi jusqu'à ce que l'on enregistre un succès suivi d'un échec dans chacune des directions. L'étape k est alors terminée et le point auquel on est arrivé devient le point de départ pour l'étape suivante, $x_0^{k+1} = x_n^k$. La direction s_1^{k+1} est choisie parallèle à $(x_0^{k+1} - x_0^k)$ et les autres directions sont orthonormées entre elles et par rapport à s_1^{k+1} par une méthode que nous allons décrire brièvement.

Lorsque l'étape k est terminée, les vecteurs des nouvelles directions de recherche doivent être calculées au point $x_0^{k+1} = x_n^k$. On procéde à une rotation des axes de recherche de manière à ce qu'ils soient alignés avec une vallée ou une crête de l'hypersurface et qu'ils éliminent les interactions entre variables.

Si d_i^k est la somme algébrique de tous les pas de recherche ayant entraîné un succès dans la direction s_i^k durant l'étape k on peut alors définir les n vecteurs $A_1, \ldots, A_n$ tels que:

$$\begin{aligned} A_1^k &= d_1^k s_1^k + d_2^k s_2^k + \ldots + d_n^k s_n^k \\ A_2^k &= \quad d_2^k s_2^k + \ldots + d_n^k s_n^k \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ A_n^k &= \quad d_n^k s_n^k \end{aligned} \qquad \text{(eq. 7)}$$

A_1^k est alors le vecteur joignant le point initial au point final obtenu par l'utilisation des vecteurs s_i^k et de la méthode de recherche directe définie précédemment.
A_2^k est la somme des progressions effectuées dans les directions autres que la première.
A_3^k est la somme des progressions effectuées dans les directions autre que la première et la deuxième, etc.

En pratique les nouvelles directions de recherche sont alors déterminées comme suit:

Le premier vecteur unité s_1^{k+1} des nouvelles directions de l'étape k+1 est aligné avec le vecteur A_1^k, direction préférentielle de recherche. Les autres directions sont alors déterminées par des vecteurs orthonormés entre eux et par rapport à A_1^k. Rosenbrock utilise pour cela la méthode d'orthogonalisation de Gram-Schmidt qui permet d'écrire:

$$s_1^{k+1} = \frac{A_1^k}{\| A_1^k \|}$$

$$B_2^k = A_2^k - [\ (A_2^k)^T\ s_1^{k+1}\ ]\ s_1^{k+1}$$

$$s_2^{k+1} = \frac{B_2^k}{\| B_2^k \|} \qquad \text{(eq. 8)}$$

$$B_2^k = A_2^k - \sum_{i=1}^{n-1} [\ (A_n^k)^T\ s_i^{k+1}\ ]\ s_i^{k+1}$$

$$s_n^{k+1} = \frac{B_n^k}{\| B_n^k \|}$$

avec $\| A_i \|$ = norme de A_i

Une procédure de recherche identique à celle de l'étape k est alors utilisée pour l'étape k+1 en partant du point $x_0^{k+1}=x_n^k$.

L'arrêt automatique de la méthode peut être envisagé au delà d'un nombre donné d'étapes de recherche ou encore lorsque les variations de la fonction critère restent inférieures à une valeur donnée.

La méthode de Rosenbrock est présentée de manière synthétique en figure 4 sous forme d'organigramme. Nous proposerons cependant l'amélioration suivante: si le nombre d'échecs successifs enregistrés dans un même système d'axes est trop important tous axes confondus (la limite est fixée ici, arbitrairement, à 5 * (nb paramètres)), on se replace alors au début de l'algorithme. Cela a pour conséquence de reprendre un système d'axes qui corresponde à celui des paramètres mais cela permet aussi de retrouver les pas de calcul initiaux. Une succession d'échecs entraîne en effet une forte réduction du pas de recherche qui gêne les possibilités de progression de la méthode.

ORGANIGRAMME METHODE DE ROSENBROCK

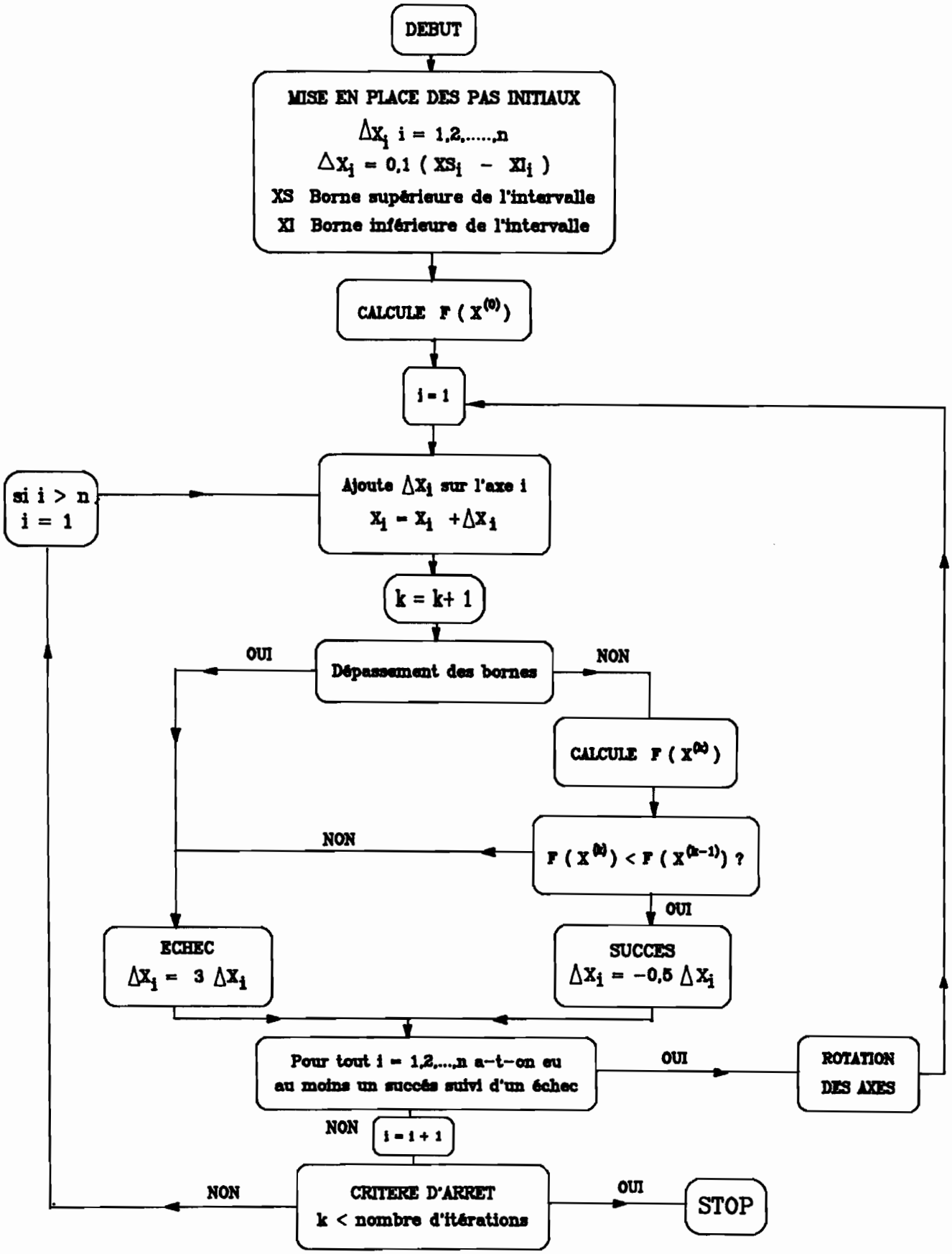


Figure 4

BIBLIOGRAPHIE

HIMMELBLAU D. 1972 "Applied non linear programming", 497 pages, Mac Graw Hill éditeurs.

NELDER J.A., MEAD R. 1964 "A simple method for function minimization", Computer Journal, 7: 308-313.

PAVIANI D. 1969 Ph D, University of Texas, Austin, Texas

ROSENBROCK H.H. 1960 "An automatic method for finding the greatest or least value of a function", Computer Journal, 3,175.

SPENDLEY W., HEXT G. R., HIMSWORTH F.R. 1962 "The sequential application of simplex designs in optimization and evolutionary operation", Technometrics, 4: 441

3. STRUCTURE ET UTILISATION DES SOUS-PROGRAMMES D'OPTIMISATION: "SIMPLE" ET "ROSEN".

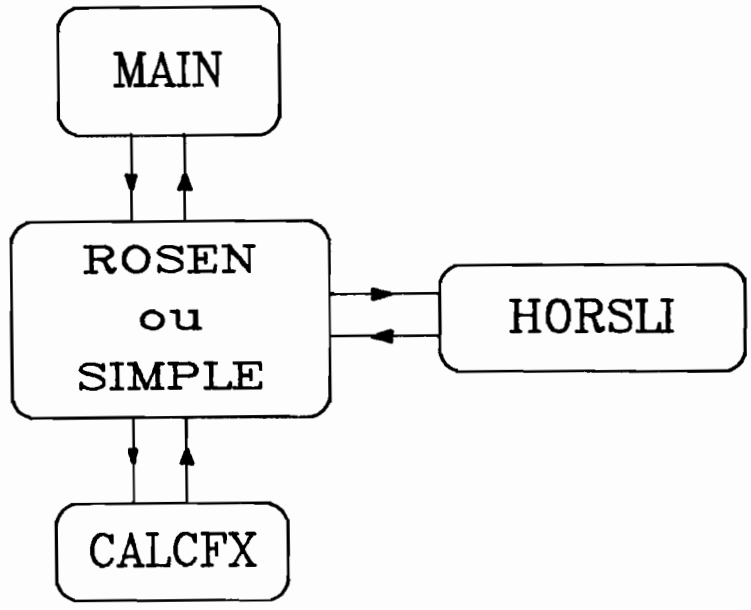
3.1 Utilisation des méthodes d'optimisation

Les sous-programmes correspondant à ces méthodes d'optimisation ont été écrits en **FORTRAN** norme **ANSI 77**. Ils sont donc actuellement utilisables sur tout ordinateur équipé d'un compilateur correspondant, sans préjuger du système d'exploitation propre à la machine. (exemple: le compilateur **FORTRAN PROFESSIONNEL** d'IBM sous MS/DOS).

La réalisation d'une optimisation pour un problème donné nécessitera la constitution d'un programme comprenant les parties suivantes:

- programme principal (**MAIN**)
- sous-programme d'optimisation (Routine **ROSEN ou SIMPLE**)
- sous-programme de vérification des paramètres (Routine **HORSLI**)
- sous-programme de calcul de la fonction critère (Routine **CALCFX**)

Les programmes correspondant aux techniques décrites dans ce document sont dénommés ROSEN pour la méthode de Rosenbrock, et SIMPLE pour la méthode de Nelder et Mead. On trouvera les listings de ces algorithmes plus loin. Outre les techniques d'optimisation proprement dites on trouvera également les sous-programmes HORSLI et un exemple de CALCFX relatif à l'optimisation des paramètres d'une fonction donnée.



MAIN et CALCFX élaborés, le fonctionnement du logiciel est le suivant:

1) MAIN: lecture des données, initialisation des paramètres et des variables

2) Appel de la procédure d'optimisation (ROSEN ou SIMPLE)

3) Développement du processus d'optimisation avec, à chaque itération (si nécessaire), vérification des paramètres par rapport à leurs bornes respectives. Si le test est concluant, on calcule la valeur de la fonction critère (CALCFX) pour chaque nouveau jeu de paramètres et on la compare à la valeur calculée à l'itération précédente. S'il y a amélioration, on stocke les valeurs correspondantes des paramètres.

4) L'optimisation terminée (critère d'arrêt satisfait ou nombre d'itérations maximum atteint), on renvoie dans le MAIN le jeu de paramètres optimum déterminé et la valeur correspondante de la fonction critère.

La tâche dévolue à l'utilisateur consiste à:

1) Elaborer un programme principal, MAIN, dans lequel on lit les données nécessaires à la fonction ou au modèle pour lesquels on cherche les valeurs optimales des paramètres (données qui seront passées dans CALCFX au moyen d'une instruction COMMON; par exemple pluie, débit, et évaporation dans le cas d'un modèle hydrologique pluie-débit). En outre, le MAIN doit permettre d'initialiser toutes les valeurs des différents arguments d'appel de ROSEN ou de SIMPLE, à savoir:

* XS, bornes supérieures des paramètres à optimiser

* XI, bornes inférieures des paramètres à optimiser

* X , valeurs des paramètres à optimiser

* IOPT, indices des paramètres à optimiser, en fonction de leur classement dans X

* ICTR, types de bornes:

  ICTR = 0, pas de borne
  ICTR = 1, borne supérieure seule
  ICTR = -1, borne inférieure seule
  ICTR = 2, borne inférieure et supérieure

* NK, nombre d'itérations

* NOPT, nombre de paramètres à optimiser

* ITIMP, affichage des résultats intermédiaires en cours d'optimisation (ITIMP = 1 --> oui, ITIMP = 0 --> non)

* ICOPT, canal logique de sortie des résultats intermédiaires

* F, valeur de la fonction critère pour le jeu de paramètres obtenu après optimisation.

Dans le cas de la méthode de Nelder et Mead (et donc de la Routine SIMPLE), il convient de rajouter un argument: STEP, pas de calcul du processus.

2) Construire une Routine CALCFX autour de la fonction ou du modèle pour lequel on cherche un jeu optimal de paramètres. En pratique, dans CALCFX, le modèle proprement dit va coexister avec la fonction critère. On aura, par exemple, dans cette Routine (1) l'ensemble des équations décrivant le modèle CREC et permettant le calcul des débits pour le jeu de paramètres considéré et (2) une fonction critère qui pourrait être dans ce cas le calcul de la somme des carrés des écarts entre les débits observés et calculés sur la période de référence. Les données seront transférées du MAIN dans CALCFX par un COMMON (voir le listing joint). Les arguments d' appel de cette Routine sont:

* N, nombre de paramètres à optimiser

* X, valeurs des paramètres à optimiser

* F, valeur de la fonction critère

3.2 Fiches techniques

SIMPLE

DATE	AUTEUR	LANGAGE	SORTIE(S)
09/05/88	A.DEZETTER	FORTRAN 77	**X** Vecteur des paramètres optimisés **F** Valeur de la fonction

1. FONCTION

Minimise une fonction F(X1,X2,.....,Xn) en utilisant la méthode de Nelder et Mead

2.APPEL

```
CALL SIMPLE(XS,XI,X,IOPT,ICTR,NK,NOPT,STEP,ITIMP,ICOPT,F)
```

3.ARGUMENTS

11 Arguments : 9 en entrée
1 en sortie (F)
1 vecteur en entrée-sortie (X)

Entrée

.Tableaux à dimensionner dans le programme appelant

```
Fa  XS    .... Bornes supérieures des paramètres          REAL*4
Fa  XI    .... Bornes inférieures des paramètres          REAL*4
0   X     .... Valeurs des paramètres au point de départ REAL*4
0   IOPT  .... Indice par rapport au vecteur X des
                 des paramètres à optimiser              INTEGER*4
0   ICTR  .... Indicateur d'utilisation ou non de        INTEGER*4
               bornes pour chaque élément du vecteur X
        peut prendre 4 valeurs :  0  pas de bornes
                                  1  borne supérieure uniquement
                                 -1  borne inférieure uniquement
                                  2  bornes inf. et sup.
```

. Autres variables à définir dans le programmme appellant

```
0   NK    .... Nombre maximum d'itérations              INTEGER*4
0   NOPT  .... Nombre de paramètres à optimiser         INTEGER*4
0   STEP  .... Valeur du pas initial du simplex            REAL*4
0   ITIMP .... Indicateur pour la sortie de résultats   INTEGER*4
               intermédiaires 1 = oui, 0 = non
0   ICOPT .... Unité logique de sortie des résultats    INTEGER*4
                                    intermédiaires
```

Sortie

F	....	Valeur de la fonction au point de coordonnées renvoyées par le vecteur X	REAL*4
X	....	Valeurs des paramètres après optimisation	REAL*4

0 : à fournir obligatoirement
Fa : Facultatif ou obligatoire selon la valeur de ICTR

4.MODULES EXTERNES

* Subroutines amont : aucune

* Subroutines appelées :

START : Initialisation de la matrice du simplex
HORSLI : Vérification des bornes des paramètres
CALCFX : Calcul de la valeur de la fonction au point de coordonnées X

5.Exemple

Exemple de MAIN et de routine CALCFX à batir pour l'utilisation de la méthode de Nelder et Mead
A. DEZETTER - 04 - 88 -

```
      COMMON /DONNEES/Y1(20),Y2(20)
      REAL XS(50),XI(50),X(50)
      INTEGER IOPT(50),ICTR(50),N,NOPT,I
      DO 10 I=1,50
      XS(I)=0.
      XI(I)=0.
      X(I)=0.
      IOPT(I)=0
10    ICTR(I)=0
Remplissage des variables du common données
      DO 9 I=1,20
      Y1(I)=I
      Y2(I)=2*I
 9    CONTINUE
      WRITE(*,11)
11    FORMAT(1X,' NOMBRE DE PARAMETRES A OPTIMISER ')
      READ(*,*)NOPT
      DO 20 I=1,NOPT
      WRITE(*,12)
12    FORMAT(1X,' INDICE DU PARAM BORNE INF BORNE SUP ')
      READ(*,*)IOPT(I),XI(I),XS(I)
      WRITE(*,13)
13    FORMAT(1X,' VALEUR DE DEPART ')
      READ(*,*)X(I)
20    CONTINUE
      DO 30 I=1,50
30    ICTR(I)=2
      WRITE(*,14)
14    FORMAT(1X,' NOMBRE D ITERATIONS ')
      READ(*,*)NK
```

```
      WRITE(*,15)
15    FORMAT(1X,' STEP = ')
      READ(*,*)STEP
      WRITE(*,16)
16    FORMAT('  résultats intermédiaires  tapez 1 sinon 0')
      READ(*,*)ITIMP
      ICOPT=6
      CALL SIMPLE(XS,XI,X,IOPT,ICTR,NK,NOPT,STEP,ITIMP,ICOPT,F)
      WRITE(*,9011)NK,(X(IOPT(I)),I=1,NOPT)
9011  FORMAT(1X,/,1X,'SOLUTION NELDER APRES',I10,'ITERATIONS',//
     *,' VALEUR DES PARAMETRES:',12(F9.4),//)
      WRITE(*,9021)F
9021  FORMAT(1X,'VALEUR FONCTION CRITERE= ',F12.4)
      STOP
      END

      SUBROUTINE CALCFX(N,X,F)
      COMMON /DONNEES/Y1(20),Y2(20)
      INTEGER I,N
      REAL X(1),F,S
      S=0.
      DO 20 I=1,20
      S=S+ X(1)*Y1(I)*Y1(I) + X(2)*Y2(I) + X(3)
20    CONTINUE
      F=S
      RETURN
      END
```

6.Traitement des erreurs

Message d'erreur à l'écran si le point de départ est hors de l'intervalle précisé par les bornes des paramètres

7.Remarques

8.Votre implantation

Type de machine : toute machine ayant un compilateur Fortran 77
Système d'exploitation : indifférent.

Actuellement a été testé sous Ms-Dos avec le compilateur IBM Professionnel et sous Aegis - Unix sur station de travail Apollo

ROSEN

DATE	AUTEUR	LANGAGE	SORTIE(S)
09/05/88	A.DEZETTER	FORTRAN 77	**X** Vecteur des paramètres optimisés **F** Valeur de la fonction

1. FONCTION

Minimise une fonction F(X1,X2,.....,Xn) en utilisant la méthode de Rosenbrock

2.APPEL

```
CALL ROSEN(XS,XI,X,IOPT,ICTR,NK,NOPT,ITIMP,ICOPT,F)
```

3.ARGUMENTS

11 Arguments : 8 en entrée
1 en sortie (F)
1 vecteur en entrée-sortie (X)

Entrée

.Tableaux à dimensionner dans le programme appelant

```
Fa  XS    .... Bornes supérieures des paramètres          REAL*4
Fa  XI    .... Bornes inférieures des paramètres          REAL*4
0   X     .... Valeurs des paramètres au point de départ REAL*4
0   IOPT  .... Indice par rapport au vecteur X des
                 des paramètres à optimiser             INTEGER*4
0   ICTR  .... Indicateur d'utilisation ou non de       INTEGER*4
               bornes pour chaque élément du vecteur X
      peut prendre 4 valeurs :  0  pas de bornes
                                1  borne supérieure uniquement
                               -1  borne inférieure uniquement
                                2  bornes inf. et sup.
```

. Autres variables à définir dans le programmme appellant

```
0   NK    .... Nombre maximum d'itérations              INTEGER*4
0   NOPT  .... Nombre de paramètres à optimiser         INTEGER*4
0   ITIMP .... Indicateur pour la sortie de résultats INTEGER*4
               intermédiaires 1 = oui, 0 = non
0   ICOPT .... Unité logique de sortie des résultats  INTEGER*4
                                      intermédiaires
```

Sortie

F	....	Valeur de la fonction au point de coordonnées renvoyées par le vecteur X	REAL*4
X	....	Valeurs des paramètres après optimisation	REAL*4

0 : à fournir obligatoirement
Fa : Facultatif ou obligatoire selon la valeur de ICTR

4.MODULES EXTERNES

* Subroutines amont : aucune

* Subroutines appelées :

HORSLI : Vérification des bornes des paramètres
CALCFX : Calcul de la valeur de la fonction au point de coordonnées X

5.Exemple

Exemple de MAIN et de routine CALCFX à batir pour l'utilisation de la méthode de Rosenbrock

A. DEZETTER - 04 - 88 -

```
      COMMON /DONNEES/Y1(20),Y2(20)
      REAL XS(50),XI(50),X(50)
      INTEGER IOPT(50),ICTR(50),N,NOPT,I
      DO 10 I=1,50
      XS(I)=0.
      XI(I)=0.
      X(I)=0.
      IOPT(I)=0
10    ICTR(I)=0
      DO 9 I=1,20
      Y1(I)=I
      Y2(I)=2*I
 9    CONTINUE
      WRITE(*,11)
11    FORMAT(1X,' NOMBRE DE PARAMETRES A OPTIMISER ')
      READ(*,*)NOPT
      DO 20 I=1,NOPT
      WRITE(*,12)
12    FORMAT(1X,' INDICE DU PARAM BORNE INF BORNE SUP ')
      READ(*,*)IOPT(I),XI(I),XS(I)
      WRITE(*,13)
13    FORMAT(1X,' VALEUR DE DEPART ')
      READ(*,*)X(I)
20    CONTINUE
      DO 30 I=1,50
30    ICTR(I)=2
      WRITE(*,14)
14    FORMAT(1X,' NOMBRE D ITERATIONS ')
      READ(*,*)NK
      WRITE(*,16)
16    FORMAT('  résultats intermédiaires  tapez 1 sinon 0')
      READ(*,*)ITIMP
      ICOPT=6
```

```
      CALL ROSEN (XS,XI,X,IOPT,ICTR,NK,NOPT,ITIMP,ICOPT,F)

      WRITE(*,9010)NK,(X(IOPT(I)),I=1,NOPT)
9010  FORMAT(1X,/,1X,'SOLUTION ROSENBROCK APRES ',I10,' ITERATIONS',//,5
     *X,'VALEUR DES PARAMETRES:',12(F9.4),//)
      WRITE(*,9020)F
9020  FORMAT(1X,'VALEUR FONCTION CRITERE= ',F12.4)
      STOP
      END

      SUBROUTINE CALCFX(N,X,F)
      COMMON /DONNEES/Y1(20),Y2(20)
      INTEGER I,N
      REAL X(1),F,S
      S=0.
      DO 20 I=1,20
      S=S+ X(1)*Y1(I)*Y1(I) + X(2)*Y2(I) + X(3)
20    CONTINUE
      F=S
      RETURN
      END
```

6.Traitement des erreurs

7.Remarques

8.Votre implantation

Type de machine : toute machine ayant un compilateur Fortran 77
Système d'exploitation : indifférent.

Actuellement a été testé sous Ms-Dos avec le compilateur IBM Professionnel et sous Aegis - Unix sur station de travail Apollo

3.3 Exemples d'utilisation

L'exemple ci-dessous utilise la routine CALCFX présentée dans les fiches techniques. Il correspond à la minimisation de la fonction $F(x_1,x_2,x_3)$ définie par :

$$F(x_1,x_2,x_3) = x_1 * \sum_{i=1}^{i=20} y_1(i)^2 + x_2 \sum_{i=1}^{i=20} y_2(i) + x_3$$

où
$y_1(i)=i$ et $y_2(i)=2*i$ pour i=1,20

La solution triviale est bien entendu $x_1 = x_2 = x_3 = 0$

Les tableaux ci-après présentent les résultats obtenus par les deux méthodes pour le même point de départ et pour différentes séries d'itérations

Valeur de la fonction au point de départ F = 6620
Au départ :
X1 = 2 XS = 0 XI = 10
X2 = 2 XS = 0 XI = 10
X3 = 2 XS = 0 XI = 10

Nelder et Mead

Nombre d'itérations	50	100	200	500	1000	10000
F	50.96241379	49.25020218	0.87141675	0.82213843	0.82213843	0.82213843
X1	0.00035426	0.00000183	0.00028693	0.00028546	0.00028546	0.00028546
X2	0.00111435	0.00000525	0.00003856	0.00000100	0.00000100	0.00000100
X3	2.47388363	2.46213770	0.00158610	0.00012284	0.00012284	0.00012284

Rosenbrock

Nombre d'itérations	50	100	200	500	1000	10000
F	581.73352051	57.20058060	10.64296627	10.64296627	10.64296627	10.6429662
X1	0.12182024	0.00137484	0.00088656	0.00088656	0.00088656	0.00088656
X2	0.52437460	0.10783482	0.01408482	0.01408482	0.01408482	0.01408482
X3	0.59360290	0.39820838	0.10914588	0.10914588	0.10914588	0.10914588

Dans ce cas particulier, on remarquera qu'au delà de 200 itérations la méthode de Rosenbrock ne progresse plus. La méthode de Nelder et Mead est, quant à elle, bloquée à partir de 500 itérations. Les résultats fournis par cette dernière méthode étant les meilleurs tant en terme de fonction critère que de valeurs de paramètres. Nous n'en conclurons pas pour autant que la méthode du Simplex est toujours plus performante que la méthode de Rosenbrock. L'une ou l'autre de ces techniques d'optimisation se révélera mieux adaptée selon le cas traité: conditions initiales, nature de la fonction à optimiser et donc forme de l'hypersurface, etc. C'est l'expérience qui guidera l'utilisateur dans ses choix et dans ses conclusions.

```
C-----------------------------------------------------------------------C
C   SUBROUTINE SIMPLE                                                   C
C                                                                       C
C     - APPELLE    CALCFX :  calcul de la fonction à minimiser          C
C                  HORSLI :  vérification des bornes                    C
C                  START  :  mise en place de la matrice du simplex     C
C                                                                       C
C-----------------------------------------------------------------------C
C    VERSION ORIGINALE    -  APPLIED NON LINEAR PROGRAMMING -           C
C                             D. HIMMELBLAU  MC GRAW HILL               C
C                                                                       C
C    MODIFIEE  - ?? 87 -  C. BOUVIER                                    C
C                                                                       C
C    ****************************************                           C
C    SOUS PROGRAMME OPTIMISATION NELDER ET MEAD                         C
C    ****************************************                           C
C                                                                       C
C    VERSION ADAPTEE ET CORRIGEE PAR C.BOUVIER                          C
C                                                                       C
C                                                                       C
C    + Declaration de tous les reels en double precision :              C
C      absolument obligatoire des que le critere arret                  C
C      est inferieur a 1.E-07 .....................10/87                C
C                                                                       C
C    + Correction erreur dans le programme :                            C
C      instruction   IF((INDEX-1).EQ.0) GO TO 12 remplace               C
C      par           IF((INDEX-I).EQ.0) GO TO 12.....11/87              C
C                                                                       C
C    + Introduction de la variable LOP permettant de                    C
C      selectionner les parametres sur lesquels doit porter             C
C      la minimisation.............................11/87                C
C                                                                       C
C                                                                       C
C     MODIFIEE  - 03 88 -  A. DEZETTER                                  C
C                                                                       C
C  But des modifications, homogénéiser avec la méthode de Rosenbrock    C
C                                                                       C
C                                                                       C
C    +  Ajout de la routine horsli qui renvoie lh=1 si un paramètre     C
C       est hors limites                                                C
C                                                                       C
C    +  changement des arguments de SIMPLE pour être homogène avec      C
C       Rosenbrock, utilisation des mêmes variables: X,XS,XI,IOPT,ICTR C
C       NOPT,NK  seul STEP spécifique à Nelder est en plus par rapport C
C       à Rosenbrock                                                    C
C                                                                       C
C    +  ajout de l'argument ITIMP et ICOPT                              C
C       ITIMP = 1 écriture de résultats intermédiaires                  C
C       ITIMP = 0 pas de résultats intermédiaires                       C
C       ICOPT canal logique de sortie des résulats intermédiaires       C
C                                                                       C
C    +  Ajout de F dans les arguments, renvoie la valeur de             C
C       la fonction optimisée pour le jeu de paramètres X               C
C       renvoyé                                                         C
C                                                                       C
C                                                                       C
C                                                                       C
C   SUML : Valeur minimum de la fonction a optimiser                    C
```

```
C   SUMH : Valeur maximum de la fonction a optimiser                  C
C   SUMS : 2e plus grande valeur de la fonction a optimiser           C
C   K1   : indice du point supplementaire destine a construire le     C
C          simplex initial                                            C
C   K2   : indice du centre de gravite des points avec indice i       C
C          different de h                                             C
C   K3   : indice du point reflexion                                  C
C   K4   : indice du point contraction ou expansion                   C
C   INDEX: indice du point correspondant a SUMH                       C
C   KOUNT: indice du point correspondant a SUML                       C
C                                                                     C
C                                                                     C
C                                                                     C
C                                                                     C
C-------------------------------------------------------------------C


      SUBROUTINE SIMPLE (XS,XI,X,IOPT,ICTR,NK,NOPT,STEP,ITIMP,ICOPT,F)



C     NOPT=nombre de parametres
C     STEP=pas de calcul
C     X=vecteur des parametres
C     XI vecteur des bornes inférieures
C     XS vecteur des bornes supérieures
C     IOPT vecteur indice, par rapport au vect X, des paramètres à
C     optimiser
C     ICTR = 0 pas de limite
C     ICTR = 1 limite sup seule
C     ICTR = -1 limite inf seule
C     ICTR = 2 limite inf et sup
C     SUM=fonction critere
C     X1=vecteur de travail


      REAL*4 X1(50,50),X(1),XS(1),XI(1),SUM(50),STEP,F
      REAL*4 ALFA,BETA,GAMA,DIFER,SUMH,SUML,SUM2,SUMS
      INTEGER*4 IOPT(1),ICTR(1),ITIMP,ICOPT,NK,NOPT
      INTEGER*4 KR,LH,K1,K2,K3,K4,I,J,INDEX,KOUNT
      KR=0
      ALFA=1.0
      BETA=0.5
      GAMA=2.0
      DIFER= 0.
      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)
      IF (LH.EQ.0) THEN
          CALL CALCFX(NOPT,X,F)
          SUM(1)=F
       ELSE
       WRITE(*,100)
100   FORMAT('  point de départ hors limite !!!!'/)
      STOP 100
      ENDIF
      K1 = NOPT + 1
```

```
      K2 = NOPT + 2
      K3 = NOPT + 3
      K4 = NOPT + 4
      CALL START(NOPT,STEP,X,X1,IOPT)
   25 DO 3 I = 1, K1
      DO 4 J = 1, NOPT
    4 X(IOPT(J)) = X1(I ,J)
      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)
      IF (LH.EQ.0) THEN
          CALL CALCFX(NOPT,X,F)
          SUM(I)=F
       ELSE
       WRITE(*,200)
200    FORMAT('  des points initiaux du simplex sont hors limites '/
     + '  Changez la valeur du point de départ où changez la valeur de S
     +TEP'/)
       STOP 200
       ENDIF
    3 CONTINUE
C
C  Recherche de SUMH , plus forte valeur du simplex
C
   28 SUMH = SUM(1)
      KR=KR+1
      INDEX = 1
      DO 7 I = 2, K1
      IF(SUM(I).LE.SUMH) GO TO 7
      SUMH = SUM(I)
      INDEX = I
    7 CONTINUE
C
C  Recherche de SUML, plus faible valeur du simplex
C
      SUML = SUM(1)
      KOUNT = 1
      DO 8 I = 2, K1
      IF(SUML.LE.SUM(I)) GO TO 8
      SUML = SUM(I)
      KOUNT = I
    8 CONTINUE
C
C  Calcul des coordonnées du "centroïd" sans prendre en compte INDEX
C
      DO 9 J = 1, NOPT
      SUM2 = 0.
      DO 10 I = 1, K1
   10 SUM2 = SUM2 + X1(I,J)
      X1(K2,J) =1./FLOAT(NOPT)*(SUM2 - X1(INDEX,J))
C  Calcul des coordonnées de la reflexion de INDEX/centroïd
      X1(K3,J) = (1. + ALFA)*X1(K2,J) - ALFA*X1(INDEX,J)
    9 X(IOPT(J)) = X1(K3,J)
      IF (ITIMP.EQ.1) WRITE(ICOPT,2004)
2004  FORMAT('   réflection   ')
      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)
      IF (LH.EQ.0) THEN
          CALL CALCFX(NOPT,X,F)
          SUM(K3)=F
       ELSE
```

```
       SUM(K3)=SUMH+1
      ENDIF
      IF(SUM(K3).LT.SUML) GO TO 11
C
C  Recherche de la deuxième plus forte valeur du simplex
C
      IF(INDEX.EQ.1) GO TO 38
      SUMS = SUM(1)
      GO TO 39
   38 SUMS = SUM(2)
   39 DO 12 I = 1, K1
      IF((INDEX - I).EQ.0) GO TO 12
      IF(SUM(I).LE.SUMS) GO TO 12
      SUMS = SUM(I)
   12 CONTINUE
      IF(SUM(K3).GT.SUMS) GO TO 13
      GO TO 14
C
C  Calcul de l'extension si la réflexion a produit un minimum
C
11    IF (ITIMP.EQ.1) WRITE(ICOPT,2001)
2001  FORMAT('   extension   ')
      DO 15 J = 1, NOPT
      X1(K4,J) = (1 - GAMA)*X1(K2,J) + GAMA*X1(K3,J)
   15 X(IOPT(J)) = X1(K4,J)
      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)
      IF (LH.EQ.0) THEN
          CALL CALCFX(NOPT,X,F)
          SUM(K4)=F
       ELSE
       SUM(K4)=SUMH+1
      ENDIF
      IF(SUM(K4).LT.SUML) GO TO 16
      GO TO 14
   13 IF(SUM(K3).GT.SUMH) GO TO 17
      DO 18 J = 1, NOPT
   18 X1(INDEX,J) = X1(K3,J)
   17 DO 19 J = 1, NOPT
      X1(K4,J) = BETA*X1(INDEX,J) + (1. - BETA)*X1(K2,J)
   19 X(IOPT(J)) = X1(K4,J)
      IF (ITIMP.EQ.1) WRITE(ICOPT,2003)
2003  FORMAT('   réduction   ')

      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)
      IF (LH.EQ.0) THEN
          CALL CALCFX(NOPT,X,F)
          SUM(K4)=F
       ELSE
       SUM(K4)=SUMH+1
      ENDIF
      IF(SUMH.GT.SUM(K4)) GO TO 16
C
C  Réduit le simplex de moitié si la reflexion a produit une valeur plus
C  forte que le maximum
C
      IF (ITIMP.EQ.1) WRITE(ICOPT,2002)
2002  FORMAT('    contraction   ')
      DO 20 J = 1, NOPT
```

```
      DO 20 I = 1, K1
   20 X1(I,J) = 0.5*(X1(I,J) + X1(KOUNT,J))
      DO 29 I = 1, K1
      DO 30 J = 1, NOPT
   30 X(IOPT(J)) = X1(I,J)
      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)
      IF (LH.EQ.0) THEN
          CALL CALCFX(NOPT,X,F)
          SUM(I)=F
       ELSE
       SUM(I)=SUMH+1
      ENDIF
   29 CONTINUE
      GO TO 26
   16 DO 21 J = 1, NOPT
      X1(INDEX,J) = X1(K4,J)
   21 X(IOPT(J)) = X1(INDEX,J)
      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)
      IF (LH.EQ.0) THEN
          CALL CALCFX(NOPT,X,F)
          SUM(INDEX)=F
       ELSE
       SUM(INDEX)=SUMH+1
      ENDIF
      GO TO 26
   14 DO 22 J = 1, NOPT
      X1(INDEX,J) = X1(K3,J)
   22 X(IOPT(J)) = X1(INDEX,J)
      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)
      IF (LH.EQ.0) THEN
          CALL CALCFX(NOPT,X,F)
          SUM(INDEX)=F
       ELSE
       SUM(INDEX)=SUMH+1
      ENDIF
   26 DO 23 J = 1, NOPT
   23 X(IOPT(J)) = X1(K2,J)
      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)
      IF (LH.EQ.0) THEN
          CALL CALCFX(NOPT,X,F)
          SUM(K2)=F
       ELSE
       SUM(K2)=SUMH+1
      ENDIF
      DIFER = 0.
      DO 24 I = 1, K1
   24 DIFER = DIFER + (SUM(I) - SUM(K2))**2
      DIFER =SQRT(DIFER)/FLOAT(NOPT)
      IF (ITIMP.EQ.1) WRITE(ICOPT,101)KR,SUML,(X1(KOUNT,J),J=1,NOPT)
  101 FORMAT(' itération : ',I6,' SUML = ',E16.6,/,' Valeur des paramètr
     +es  '/,5E16.6,(/,5E16.6))
      IF (ITIMP.EQ.1) WRITE(ICOPT,102)DIFER
102   FORMAT('  DIFER (critère d''arrêt, arrêt si <1E-6) ',E16.6)
      IF( DIFER.GE.1.E-6.AND.KR.LT.NK) GO TO 28
      DO 70 I=1,NOPT
70    X(IOPT(I))=X1(KOUNT,I)
      F=SUM(KOUNT)
      END
```

```
C !!!
C !!!
      SUBROUTINE START(NOPT,STEP,X,X1,IOPT)
C CALCUL DE STEP1 ET STEP2
C MISE EN PLACE DE LA MATRICE DU SIMPLEX

      REAL*4 A(50,50),X1(1,1),X(1),STEP,STEP1,STEP2,VN
      INTEGER*4 NOPT,IOPT(1),I,J,L
      VN = NOPT
      STEP1 = STEP/(VN*SQRT(2.))*(SQRT(VN + 1.) + VN - 1.)
      STEP2= STEP/(VN*SQRT(2.))*(SQRT(VN + 1.) - 1.)
      DO 1 J = 1, NOPT
    1 A(1,J) = 0.
      DO 2 I = 2, NOPT + 1
      DO 2 J = 1, NOPT
      A(I,J) = STEP2
      L = I - 1
      A(I,L) = STEP1
    2 CONTINUE
      DO 3 I = 1, NOPT + 1
      DO 3 J = 1, NOPT
    3 X1(I,J) = X(IOPT(J)) + A(I,J)
      RETURN
      END


      SUBROUTINE HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)

      REAL*4 X(1),XI(1),XS(1)
        INTEGER*4 IOPT(1),ICTR(1),NOPT,LH,I
      LH = 0
      DO 10 I=1,NOPT
      IF (ICTR(IOPT(I)).NE.0) THEN
         IF (ICTR(IOPT(I)).EQ.-1) THEN
         IF ((X(IOPT(I))-XI(IOPT(I))).LE.1E-6) GOTO 30
         ENDIF
         IF (ICTR(IOPT(I)).EQ.1) THEN
         IF ((X(IOPT(I))-XS(IOPT(I))).GE.-1E-6) GOTO 30
         ENDIF
         IF (ICTR(IOPT(I)).EQ.2) THEN
         IF ((X(IOPT(I))-XI(IOPT(I))).LE.1E-6) GOTO 30
         IF ((X(IOPT(I))-XS(IOPT(I))).GE.-1E-6) GOTO 30
         ENDIF
      ENDIF
10    CONTINUE
      GOTO 40
30    LH = 1
40    RETURN
      END
```

```
C=====================================================================C
C                                                                     C
C          SUBROUTINE ROSEN                                           C
C                                                                     C
C           - APPELLE  CALCFX  : calcul de la fonction a optimiser    C
C                      HORSLI  : vérification des limites             C
C                                                                     C
C=====================================================================C
C                                                                     C
C     VERSION ORIGINALE  -- ? ? ?  --                                 C
C                                                                     C
C            Modifiee  -  01 88 -  A. DEZETTER                        C
C                                                                     C
C             -  Pas E(J) proportionnels a la taille de l'intervalle  C
C                                                                     C
C             -  reecriture de facon moins "imbriquee"                C
C                                                                     C
C             -  ajout de NECHEC :  si NECHEC >  5 *  NOPT  on        C
C          repart avec les axes de depart (on recommence une serie    C
C          d'iterations)                                              C
C                                                                     C
C                                                                     C
C            Modifiee  - 03 88  -  A. DEZETTER                        C
C                                                                     C
C             -   Ajout de la subroutine HORSLI (id NELDER), renvoie  C
C                 lh = 1 si un des paramètres est hors limites        C
C             Attention : un paramètre égal une borne est considéré   C
C             comme hors limite                                       C
C                                                                     C
C             - ajout de l'argument ITIMP et ICOPT                    C
C            ITIMP = 1 écriture de résultats intermédiaires           C
C            ITIMP = 0 pas de résultats intermédiaires                C
C            ICOPT canal logique de sortie des résulats intermédiaireC
C                                                                     C
C             - Ajout de F dans les arguments, renvoie la valeur de   C
C               la fonction optimisée pour le jeu de paramètres X     C
C               renvoyé                                               C
C                                                                     C
C        Modifiee  -      -                                           C
C                                                                     C
C              -                                                      C
C                                                                     C
C              -                                                      C
C                                                                     C
C              -                                                      C
C                                                                     C
C                                                                     C
C                                                                     C
C=====================================================================C
C                                                                     C
C  ARGUMENTS                                                          C
C  ---------                                                          C
C                                                                     C
C                                                                     C
C     NOPT=NOMBRE DE PARAMETRES A OPTIMISER                           C
C     NK=NOMBRE D'ITERATIONS                                          C
C     X=VALEURS DES PARAMETRES A OPTIMISER                            C
C     XS=BORNES SUPERIEURES DES PARAMETRES                            C
```

```
C     XI=BORNES INFERIEURES DES PARAMETRES                              C
C     IOPT=INDICE DES PARAMETRES A OPTIMISER                            C
C     ICTR=1 LIMITE SUP SEULE                                           C
C     ICTR=-1 LIMITE INF SEULE                                          C
C     ICTR=2 LIMITE SUP ET INF                                          C
C     ICTR=0 pas de limite                                              C
C                                                                       C
C=======================================================================C

      SUBROUTINE ROSEN(XS,XI,X,IOPT,ICTR,NK,NOPT,ITIMP,ICOPT,F)


      REAL*4 A(50,50),B(50,50),C(50,50),V(50,50)
      REAL*4 C3(50),A3(50),E(50),D(50)
      REAL*4 XS(1),XI(1),X(1),F
      REAL*4 F1,FO,A2,A4,ALFA1,ALFA2,ALFA3,ALFA,B2,C2
      INTEGER*4 IOPT(1),ICTR(1),NK,NOPT,ITIMP,ICOPT
      INTEGER*4 N,K,LH,NECHEC,J,I,IROT,L,L1,J3,K1,J2,J1


      N=NOPT
      K=0
      LH=0
      CALL CALCFX(N,X,F1)
      FO=F1
      WRITE(*,1)K,FO
    1 FORMAT(1X,'ITERATION=',I5,3X,'FO=',F12.4)
101   NECHEC=0
      DO 2 J=1,N
      DO 2 I=1,N
    2 V(I,J)=0.
      DO 3 J=1,N
      I=J
    3 V(I,J)=1.
C
C
      DO 3001 J=1,N
      E(J)=.1
      IF (ICTR(IOPT(J)).EQ.2) E(J)=.1 * (XS(IOPT(J))-XI(IOPT(J)))
      A3(J)=2.
      D(J)=0.
3001  CONTINUE
5000  J=1
4000  K=K+1
      IROT=0
      DO 4001 L=1,N
      L1=IOPT(L)
4001  X(L1)=X(L1)+E(L)*V(L,J)

      CALL HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)

      IF(LH.EQ.0) THEN
         CALL CALCFX(N,X,F1)
         IF(F1.LT.FO) THEN
           D(J)=D(J)+E(J)
           E(J)=3*E(J)
           FO=F1
           IF (ITIMP.EQ.1) WRITE(ICOPT,1)K,FO
```

```
          NECHEC=0
          IF (ITIMP.EQ.1) WRITE(ICOPT,1013)
1013      FORMAT('  SUCCES ')
          IF(A3(J).EQ.2) A3(J)=1
        ELSE
          IF (ITIMP.EQ.1) WRITE(ICOPT,1)K,FO
          NECHEC=NECHEC+1
          IF (ITIMP.EQ.1) WRITE(ICOPT,1011)
1011      FORMAT('  ECHEC ')
        ENDIF
      ELSE
        IF (ITIMP.EQ.1) WRITE(ICOPT,1)K,FO
        NECHEC=NECHEC+1
        IF (ITIMP.EQ.1) WRITE(ICOPT,1012)
1012    FORMAT(' HORS LIMITE ')
      ENDIF
      IF (ITIMP.EQ.1) WRITE(ICOPT,2000)(X(iopt(i)),i=1,nopt)
2000  FORMAT(' valeur des paramètres :'/(6f10.5))
      IF ((LH.EQ.1).OR.(F1.GT.FO)) THEN
        LH=0
        DO 1010 L=1,N
        L1=IOPT(L)
1010    X(L1)=X(L1)-E(L)*V(L,J)
        E(J)=-(E(J)/2.)
        IF(A3(J).EQ.1) A3(J)=0
      ENDIF
      DO 4690 J3=1,N
4690  IROT=IROT+A3(J3)


C ===================================================================
C
C    ROTATION DES AXES
C

      IF (IROT.EQ.0) THEN
        A2=0.
        IF (ITIMP.EQ.1) WRITE(ICOPT,4740)
4740    FORMAT(1X,//,1X,'ROTATION DES AXES',/)
        K1=1
4749    DO 4751 I=1,N
        DO 4750 J=K1,N
        A2=A2+ D(J)*V(I,J)
4750    CONTINUE
        A(I,K1)=A2
        A2=0.
4751    CONTINUE
        K1=K1+1
        IF(K1.LE.N) GO TO 4749
        K1=1
        A4=0
        DO 4760 I=1,N
        A4=A4 + A(I,K1)*A(I,K1)
4760    CONTINUE
        A4=SQRT(A4)
        ALFA1=A4
        IF (ITIMP.EQ.1) WRITE(ICOPT,4801)ALFA1
4801    FORMAT(1X,/,1X,'ALPHA1 =',F9.4)
```

```
      K1=K1+1
      IF (N.GT.2) THEN
         A4=0
         DO 4761 I=1,N
         A4=A4 + A(I,K1)*A(I,K1)
4761     CONTINUE
         A4=SQRT(A4)
         ALFA2=A4
         K1=K1+1
         A4=0
         DO 4762 I=1,N
         A4=A4 + A(I,K1)*A(I,K1)
4762     CONTINUE
         A4=SQRT(A4)
         ALFA3=A4
         ALFA=ALFA2/ALFA3
         IF (ITIMP.EQ.1) WRITE(ICOPT,4800)ALFA2,ALFA3,ALFA
4800     FORMAT(1X,/,1X,5X,'ALPHA2 =',F9.4,/,1X,
     *   'ALPHA3 =',F9.4,5X,'ALPHA2/ALPHA3 = ',F9.4)
      ENDIF
      J=1
      B2=0
      DO 4815 I=1,N
      B(I,J)=A(I,J)
4815  B2=B2 + B(I,J)*B(I,J)
      B2=SQRT(B2)
      DO 4817 I=1,N
4817  V(I,J)=B(I,J)/B2
      C2=0.
      B2=0.
      DO 4830 J=2,N
      J2=J-1
      DO 4819 J1=1,J2
      DO 4818 I=1,N
4818  C2=C2 + A(I,J)*V(I,J1)
      C3(J1)=C2
4819  C2=0.
      DO 4821 J1=1,J2
      DO 4820 I=1,N
4820  C(I,J1)=C3(J1)*V(I,J1)
4821  CONTINUE
      DO 4822 I=1,N
4822  C(I,J)=0.
      DO 4823 I=1,N
      DO 4823 J1=1,J2
4823  C(I,J)=C(I,J)+C(I,J1)
      DO 4824 I=1,N
      B(I,J)=A(I,J)-C(I,J)
4824  B2=B2 +  B(I,J)*B(I,J)
      B2=SQRT(B2)
      DO 4825 I=1,N
4825  V(I,J)=B(I,J)/B2
      C2=0.
      B2=0.
4830  CONTINUE
      DO 3002 J=1,N
      E(J)=.1
      IF (ICTR(IOPT(J)).EQ.2) E(J)=.1 * (XS(IOPT(J))-XI(IOPT(J)))
```

```
         A3(J)=2.
         D(J)=0.
3002     CONTINUE
      ENDIF

C
C    FIN  ROTATION
C
C ================================================================
      J=J+1
      IF (NECHEC.GT.(5*N)) THEN
          IF (ITIMP.EQ.1) WRITE(ICOPT,4810)NECHEC
4810      FORMAT(1X,/,1X,'  NOMBRE D''ECHECS :',I5,' SUPERIEUR A 5 *NOP
     *T , ON RECOMMENCE UNE SERIE D''ITERATIONS')
          NECHEC=0
          GO TO 101
      ENDIF
      IF(J.LE.N) GO TO 4000
      IF (K.LT.NK)GO TO 5000
C
      F=FO
      RETURN
      END

      SUBROUTINE HORSLI(X,XI,XS,IOPT,ICTR,NOPT,LH)

      REAL*4 X(1),XI(1),XS(1)
        INTEGER*4 IOPT(1),ICTR(1),NOPT,LH,I
      LH = 0
      DO 10 I=1,NOPT
      IF (ICTR(IOPT(I)).NE.0) THEN
         IF (ICTR(IOPT(I)).EQ.-1) THEN
         IF ((X(IOPT(I))-XI(IOPT(I))).LE.1E-6) GOTO 30
         ENDIF
         IF (ICTR(IOPT(I)).EQ.1) THEN
         IF ((X(IOPT(I))-XS(IOPT(I))).GE.-1E-6) GOTO 30
         ENDIF
         IF (ICTR(IOPT(I)).EQ.2) THEN
         IF ((X(IOPT(I))-XI(IOPT(I))).LE.1E-6) GOTO 30
         IF ((X(IOPT(I))-XS(IOPT(I))).GE.-1E-6) GOTO 30
         ENDIF
      ENDIF
10    CONTINUE
      GOTO 40
30    LH = 1
40    RETURN
      END
```