RAIRO. RECHERCHE OPÉRATIONNELLE

J. VIGNES

Implémentation des méthodes d'optimisation : test d'arrêt optimal, contrôle et précision de la solution (II)

RAIRO. Recherche opérationnelle, tome 18, n° 2 (1984), p. 103-129

<http://www.numdam.org/item?id=RO_1984__18_2_103_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 18, n° 2, mai 1984, p. 103 à 129)

IMPLÉMENTATION DES MÉTHODES D'OPTIMISATION : TEST D'ARRÊT OPTIMAL, CONTRÔLE ET PRÉCISION DE LA SOLUTION (II) (*)

par J. VIGNES [1]

Abstract. — *Lorsque l'on implémente sur ordinateur des algorithmes d'optimisation, plusieurs problèmes se posent :*

— comment arrêter le processus itératif;

— comment s'assurer que la solution obtenue est informatiquement satisfaisante et la meilleure que puisse fournir la machine;

— comment conclure que la solution informatique est une approximation de la solution mathématique et quelle est sa précision.

Nous présentons en deux articles des méthodes originales permettant de résoudre ces problèmes. Le premier article, paru dans le numéro précédent de la Revue, traitait des méthodes d'implémentation. Nous présentons ici les logiciels correspondants.

Keywords: Optimisation; analyse des erreurs d'arrondi; critère d'arrêt optimal; précision de résultats d'algorithmes; validité des logiciels numériques.

Résumé. — *In implementing any optimization method on a computer, several problems arise:*

— how to break off the iterative process;

— how to determine the satisfactory computed solution and be sure the computer cannot provide a better one;

— how to tell whether or not the computed solution approximates the mathematical solution, and to what degree of accuracy.

New methods for solving these problems appeared in the preceding issue of this Journal. The corresponding software are presented here.

Mots clés : Optimization; round-off error analysis; optimal termination criterion; accuracy in the results of computations; validity of numerical software.

I. INTRODUCTION

Pour pouvoir mettre en œuvre les méthodes décrites dans la première partie de cet article, il faut pouvoir faire faire automatiquement par la machine l'analyse de la propagation des erreurs d'arrondi due à l'arithmétique virgule flottante et pouvoir évaluer, pour toute valeur issue du calcul de n'importe quelle fonctionnelle, son nombre de chiffres décimaux significatifs exacts. Il

(*) Reçu en juin 1982.

(1) Professeur à l'Institut de Programmation de l'Université Pierre-et-Marie-Curie de Paris. Conseiller Scientifique à l'Institut Français du Pétrole.

R.A.I.R.O. Recherche opérationnelle/Operations Research, 0399-0559/84/02 103 27/$ 4.70

faut donc disposer d'un logiciel de Permutation-Perturbation permettant, d'une part, de faire exécuter le calcul de toute fonctionnelle en faisant changer l'ordre des opérations et, d'autre part, d'affecter au résultat de toute opération arithmétique une valeur soit par excès soit par défaut. C'est ce logiciel et sa mise en œuvre que nous allons décrire dans le paragraphe suivant.

II. LE LOGICIEL DE PERMUTATION-PERTURBATION

II.1. La permutation

Nous avons vu que dans la pratique il suffit de permuter les formules linéaires. Ainsi, pour réaliser les permutations, c'est-à-dire changer l'ordre d'exécution des opérations, il suffit de décomposer la fonctionnelle dont on veut estimer le nombre de chiffres décimaux significatifs exacts du résultat en une somme algébrique de monômes. Ces monômes sont rangés dans un vecteur P de dimension N, N étant le nombre de monômes.

Soit F cette fonctionnelle; elle se présente sous la forme :

$$F = \sum_{i=1}^{N} P_i. \tag{1}$$

Avant chaque exécution du calcul de F, ces N monômes P_i sont rangés dans un ordre aléatoire établi par un sous-programme de génération de nombres aléatoires dans le vecteur P, comme il est montré dans la fonction PEPER donné en annexe. Ainsi, lors des trois calculs nécessaires de F pour estimer son nombre de chiffres décimaux significatifs, le calcul de la somme des monômes se fait-il chaque fois dans un ordre différent.

Exemple. — Soit à chercher le minimum de la fonction :

$$\left.\begin{aligned} F &= f_1^2 + f_2^2, \\ f_1 &= 7\,X_1^2 + 3\,X_1\,X_2 + 4\,X_1 - X_2 - 41, \\ f_2 &= 10\,X_1^2 + 4\,X_1\,X_2 + 5\,X_1 - 2\,X_2 - 56, \end{aligned}\right\} \tag{2}$$

par la méthode présentée dans la première partie, qui permet de minimiser les fonctionnelles se présentant sous la forme d'une somme de carrés.

Il est nécessaire d'écrire le sous-programme qui calcule F et grad F sous la forme présentée en annexe qui consiste à décomposer F et grad F en une somme de monômes.

II. 2. La perturbation

La perturbation consiste à fournir comme résultat de toute opération arithmétique soit sa valeur par défaut soit sa valeur par excès. Considérons l'opération arithmétique définie par :

$$R = X \,\omega\, Y, \qquad X, Y, R \in \mathbb{F}, \tag{3}$$

ω étant un opérateur informatique quelconque.

R^- représente la valeur par défaut et R^+ la valeur par excès du résultat R. Le résultat exact $r \in \mathbb{R}$, inaccessible en informatique, est toujours compris entre R^- et R^+ :

$$R^- \leqq r \leqq R^+. \tag{4}$$

Lorsque l'ordinateur travaille en arithmétique tronquée virgule flottante normalisée avec p bits de mantisse, le résultat de toute opération arithmétique est toujours R^- (par défaut). Aussi, pour obtenir R^+ doit-on ajouter 1 au dernier bit de la mantisse.

Lorsque l'ordinateur travaille en arithmétique arrondie, le résultat de toute opération arithmétique est soit par défaut soit par excès. Aussi doit-on ajouter ou retrancher 1 au dernier bit de la mantisse ou ne rien faire (ajouter 0) pour obtenir les valeurs par défaut ou par excès.

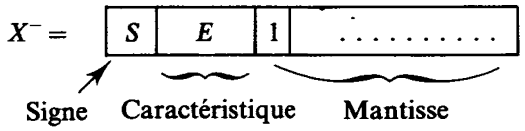
Le logiciel de perturbation consiste donc :

— dans le cas de l'arithmétique tronquée, à ajouter aléatoirement un 0 ou un 1 au dernier bit de la mantisse, les 0 et les 1 étant fournis par un générateur de nombres aléatoires uniformément répartis dans l'intervalle [0, 2[et en en prenant la partie entière;

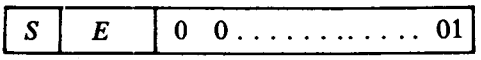
— dans le cas de l'arithmétique arrondie, à ajouter -1, 0 ou $+1$ au dernier bit de la mantisse. Les valeurs $-1, 0$ et $+1$ sont fournies par un générateur de nombres aléatoires uniformément répartis dans l'intervalle $]-2, +2[$ et en en prenant la partie entière.

Nous considérons, pour expliquer le logiciel permettant d'ajouter ou de retrancher 1 au dernier bit de la mantisse, le cas simple d'un ordinateur travaillant en arithmétique binaire normalisée avec des mots de n bits codés de la façon suivante : 1 bit pour le signe (0 pour les nombres positifs et 1 pour les négatifs), m bits réservés à la caractéristique (exposant + biais) et p bits de mantisse. Nous supposons en outre que l'arithmétique est en troncature. Ainsi, tout élément $X \in \mathbb{F}$ représentant $x \in \mathbb{R}$ représente la valeur par

défaut de x, soit X^-. Lorsqu'on ajoute un 1 au dernier bit de la mantisse, la valeur obtenue est X^+, valeur par excès de x :



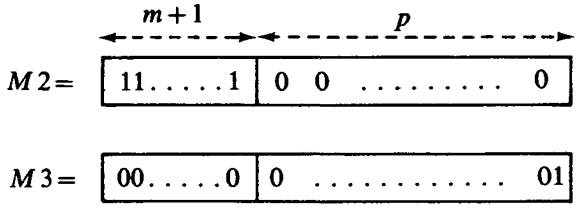
Pour obtenir X^+ il faut ajouter à X^- le nombre $M1$ défini par :



Pour obtenir $M1$ on peut écrire une procédure soit en langage d'assemblage, soit en langage Fortran qui réalise les opérations booléennes suivantes :

et
$$\left.\begin{aligned} M1 &= (X^- \cap M2) \cup M3 \\ X^+ &= X^- + M1, \end{aligned}\right\} \tag{5}$$

$M2$ et $M3$ étant les masques définis par :



Dans le cas où la codification binaire des valeurs négatives se fait en complément, il faut, pour obtenir la valeur de X^+ lorsque X^- est négatif, retrancher 1 au dernier bit de la mantisse de X^- et donc effectuer l'opération :

$$X^+ = X^- - M1. \tag{6}$$

Nous donnons en annexe la fonction XV qui permet de perturber sur CDC de la série des 6000 ou 7000 toute valeur X positive ou négative.

Le logiciel PEPER donné en annexe permet donc de réaliser pour le calcul de toute fonctionnelle la permutation et la perturbation. Ce logiciel est portable sur tout ordinateur; seule la fonction XV à laquelle il fait appel est spécifique de la machine utilisée, et doit donc être réécrite comme il a été expliqué ci-dessus.

III. ADAPTATION DU LOGICIEL PEPER AUX CODES D'OPTIMISATION

III.1. Structuration des fonctionnelles du problème à traiter

Tout code général d'optimisation nécessite que l'utilisateur écrive sous forme de procédures (sous-programmes Fortran par exemple) les fonctionnelles du problème qu'il a à traiter :

- calcul de la fonctionnelle F à optimiser,
- calcul de grad F ou $\| \text{grad } F \|$,
- calcul des contraintes.

Pour utiliser le logiciel PEPER, il est nécessaire que chacune des fonctionnelles soit écrite sous la forme d'une suite de monômes, comme il est montré dans l'exemple fourni en annexe, qui permet de résoudre le système non linéaire d'ordre 2 défini par (2) en cherchant le minimum nul de :

$$F = f_1^2 + f_2^2. \tag{7}$$

III.2. Implémentation du test d'arrêt optimal dans les codes d'optimisation de fonctions non contraintes

L'implémentation du test d'arrêt optimal dans tout code d'optimisation non contrainte est très aisée. En effet, elle consiste à remplacer le test d'arrêt classique par le calcul, à chaque itération q, du nombre de chiffres décimaux significatifs $C_i^{(q)}$ ou $K^{(q)}$ de chaque composante de grad $F(X^{(q)})$ ou de $\| \text{grad } F(X^{(q)}) \|$

Pour réaliser cela, il suffit d'appeler trois fois la procédure calculant grad $F(X^{(q)})$ ou $\| \text{grad } F(X^{(q)}) \|$ qui a été structurée comme précédemment expliqué et qui utilise le logiciel PEPER.

Ainsi obtient-on trois valeurs différentes de :

$$\frac{\partial F_j(X^q)}{\partial x_i}, \qquad i = 1, \ldots, n, \qquad j = 1, 2, 3,$$

ou de (8)

$$\| \text{grad } F_j(X^{(q)}) \|, \qquad j = 1, 2, 3.$$

On calcule ensuite la moyenne et l'écart-type de ces quantités et en utilisant l'équation (26) donnée dans la première partie de cet article, on en déduit les $C_i^{(q)}$ ou $K^{(q)}$.

Lorsque la méthode d'optimisation est convergente, les $C_i^{(q)}$, $i = 1, \ldots, n$ ou $K^{(q)}$ sont décroissants et tendent vers zéro.

Ainsi, lorsque tous les $C_i^{(q)} < 1$ $(i = 1, 2, \ldots, n)$ ou $K^{(q)} < 1$, cela veut dire alors que les composantes de grad $F(X^{(q)})$ ou $\| \text{grad } F(X^{(q)}) \|$ doivent être considérées comme nulles et qu'une solution informatiquement satisfaisante est atteinte.

Le processus itératif doit par conséquent être interrompu.

Le test d'arrêt optimal ainsi défini est présenté en annexe dans le code d'optimisation proposé.

III. 3. Implémentation du test d'arrêt optimal dans les codes d'optimisation de fonctions contraintes

Comme nous l'avons défini dans la première partie, les codes d'optimisation contrainte consistent à résoudre :

$$\left.\begin{array}{l} \underset{\text{Sup}}{\text{Inf}} \; F(X), \\ \text{sous les contraintes :} \\ \qquad H_i(X) = 0, \qquad i = 1, \ldots, m, \quad m \in \mathbb{N}, \\ \qquad G_j(X) \geqq 0, \qquad j = m+1, \ldots, p, \quad p \in \mathbb{N}. \end{array}\right\} \tag{9}$$

Comme il est montré dans la première partie de cet article, le processus itératif doit être interrompu à l'itération q lorsque $X^{(q)}$ satisfait les conditions de Kuhn et Tucker. Mais, pour s'assurer, à l'itération q, que les conditions de Kuhn et Tucker sont satisfaites, il faut que le code d'optimisation utilisé fournisse non seulement les valeurs du vecteur $X^{(q)}$ mais aussi celles des vecteurs des multiplicateurs de Lagrange $\lambda^{(q)}$ et $\mu^{(q)}$.

Pour vérifier les conditions de Kuhn et Tucker, il suffit de calculer pour tout $i = 1, \ldots, n$, en utilisant le logiciel PEPER, les nombres CL_i de chiffres décimaux significatifs de chaque composante de grad $L(X, \lambda, \mu)$, avec :

$$L(X, \lambda, \mu) = F(X) + \sum_{i=1}^{m} \mu_i H_i(X) - \sum_{j=m+1}^{p} \lambda_j G_j(X). \tag{13}$$

Aussi, si pour tout i on a :

$$CL_i < 1, \qquad i = 1, \ldots, n, \tag{14}$$

alors $X^{(q)} = X_s$ est une solution informatiquement satisfaisante dans la mesure où elle satisfait aussi les contraintes.

Pour vérifier les contraintes, il suffit de calculer le nombre de chiffres décimaux significatifs de chacune d'elles au point X_s et de s'assurer que pour

les contraintes d'égalité ces nombres sont tous inférieurs à 1 et pour les contraintes d'inégalité, que ces nombres sont supérieurs à 1.

Remarquons que lorsque les contraintes sont linéaires, on doit pour simplifier utiliser les résidus normés (*voir* première partie). Aussi, le test d'arrêt optimal permet d'arrêter le processus itératif dès que toutes les relations (14) (première partie) sont vérifiées, c'est-à-dire dès qu'une solution informatiquement satisfaisante est atteinte.

IV. ESTIMATION DE LA PRÉCISION DE LA SOLUTION

Lorsqu'un code d'optimisation contrainte ou non contrainte nous a fourni une solution X_s informatiquement satisfaisante, cela ne veut pas dire pour autant qu'elle représente une bonne approximation de la solution mathématique.

En effet, donnons de ceci une interprétation géométrique très simple sur le problème élémentaire suivant :

Trouver x^* tel que :

$$\left.\begin{array}{l} f(x^*) = \underset{\text{Nul}}{\text{Inf}}\, f(x), \qquad x \in \mathbb{R}, \\ f: \quad \mathbb{R} \Rightarrow \mathbb{R}. \end{array}\right\} \tag{15}$$

Comme il est montré dans la figure 1, chacun des points $X \in \mathbb{F}$, $X \in [X_1, X_2]$ est une solution informatique X_s satisfaisante de $F(X) = 0$.

La grandeur de l'intervalle $I_s = [X_1, X_2]$ nous permet d'évaluer le nombre de chiffres décimaux significatifs de $X_s \in \mathbb{F}$. Plus I_s est grand, moins X_s approxime précisément $x^* \in \mathbb{R}$.

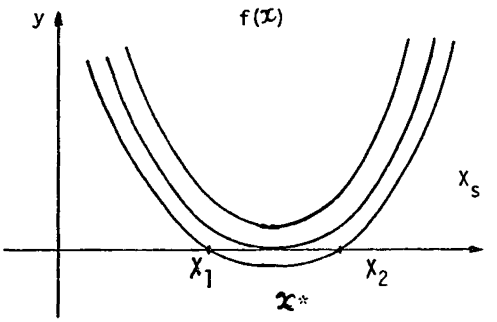


Fig. 1

Pour évaluer la précision de $X_s \in \mathbb{F}$, il suffit de résoudre trois fois le problème d'optimisation. On obtient ainsi trois solutions informatiquement satisfaisantes : $X_s^{(1)}$, $X_s^{(2)}$, $X_s^{(3)}$.

La meilleure approximation de $x^* \in \mathbb{R}$ est $\bar{X}$ moyenne des trois X_s^j, $j=1, 2, 3$. Avec ces trois résultats, il est aisé avec l'équation (26) (première partie) d'estimer le nombre de chiffres décimaux significatifs exact de $\bar{X}$.

Remarque. — Afin de minimiser le temps de calcul on peut, après avoir obtenu X_s^1, recommencer l'optimisation avec, comme nouveau point de départ, une approximation de X_s^1.

Le code donné en annexe met en œuvre cette méthode et permet donc de fournir la précision sur la solution de tout problème d'optimisation non contrainte.

V. EXEMPLES NUMÉRIQUES

V.1. Problème d'optimisation non contrainte

Nous présentons ici un exemple très simple qui permet de montrer très clairement les inconvénients de l'utilisation des tests d'arrêt classiques et l'intérêt de l'utilisation de la méthode proposée (test d'arrêt optimal et évaluation de la précision de la solution). Le logiciel correspondant à cet exemple est présenté en annexe.

Soit à résoudre le système non linéaire défini par (2) en déterminant le Inf Nul de :

$$F = f_1^2 + f_2^2. \tag{16}$$

La fonction F admet deux minima : l'un absolu, correspondant à la solution du système non linéaire considéré, l'autre relatif. Nous présentons dans le tableau I les résultats fournis par le code d'optimisation proposé en annexe.

TABLEAU I

X_1^0	X_2^0	X_{1s}	X_{2s}	CX_{1s}	CX_{2s}	F	Conclusion
−1.	50.	2.000...0	1.000...0	14	12	1.10^{-24}	X_s est solution du système linéaire
−5.	22.	−2.025386	−2.61553	7	6	5.25954	X_s n'est pas solution du système linéaire

La valeur de $F = 10^{-24}$ a été trouvée non significative et donc doit être considérée comme un zéro, et par conséquent la solution trouvée est bien solution du système non linéaire considéré.

La valeur de $F = 5.25954$ est significative et déterminée avec 12 chiffres décimaux exacts; elle ne correspond pas à un minimum nul.

Nous avons résolu le même système non linéaire en multipliant f_1 et f_2 par 10^{-20}. Les résultats obtenus sont présentés dans le tableau II.

TABLEAU II

X_1^0	X_2^0	X_{1s}	X_{2s}	CX_{1s}	CX_{2s}	F	Conclusion
−1.	50.	2.000...0	1.000...0	14	12	2.10^{-64}	X_s est solution du système non linéaire
−5.	22.	−2.025386	−2.61553	7	6	5.10^{-40}	X_s n'est pas solution du système non linéaire

Remarquons ici que le logiciel proposé a permis de conclure que la valeur $F=5.10^{-40}$ est bien significative et qu'elle est connue avec 13 chiffres décimaux significatifs. Ainsi, malgré la faible valeur de F, le logiciel a pu conclure que la solution $X_1=-2.025386$; $X_2=-2.61553$ n'est pas solution du système non linéaire.

Nous avons résolu le même système non linéaire en multipliant f_1 et f_2 par 10^{+30}. Les résultats obtenus sont présentés dans le tableau III.

TABLEAU III

X_1^0	X_2^0	X_{1s}	X_{2s}	CX_{1s}	CX_{2s}	F	Conclusion
−1.	50.	2.000...0	1.000...0	13	12	$6.6.10^{+36}$	X_s est solution du système non linéaire
−5.	22.	−2.025386	−2.61553	7	6	$5.2.10^{+61}$	X_s n'est pas solution du système non linéaire

Remarquons ici que le logiciel proposé a permis de conclure que la valeur $F=6.6.10^{+36}$ est non significative et ne représente que les erreurs de calculs; elle doit être considérée comme un zéro mathématique. Ainsi, malgré la valeur élevée de F, la solution $X_1=2$, $X_2=1$ est bien solution du système non linéaire.

Le logiciel de Permutation-Perturbation permet aussi d'évaluer l'influence des erreurs de données sur les résultats d'algorithmes numériques.

En effet il suffit, lors des trois résolutions du problème d'optimisation, de perturber les données en fonction de la précision avec laquelle elles sont connues.

Aussi nous avons résolu le même système linéaire en supposant que les valeurs des constantes qui figurent dans les fonctions f_1 et f_2 sont connues

avec une précision relative de 10^{-5}. Les résultats obtenus sont présentés dans le tableau IV.

TABLEAU IV

X_1^0	X_2^0	X_{1s}	X_{2s}	CX_{1s}	CX_{2s}	F	Conclusion
−1.	50.	2.000	1.00	4	3	$7.2.10^{-26}$	X_s est solution du système non linéaire
−5.	22.	−2.025	−2.615	4	4	5.2	X_s n'est pas solution du système non linéaire

Nous voyons ici que compte tenu des erreurs sur les données, X_1 et X_2 ne peuvent être déterminés qu'avec 3 ou 4 chiffres décimaux significatifs.

Ces exemples mettent bien en évidence que le seul test d'arrêt efficace est celui proposé ici.

Le logiciel que nous proposons et qui est donné en annexe nous permet donc d'arrêter le processus itératif dès qu'une solution informatiquement satisfaisante est atteinte, il minimise ainsi le nombre des itérations et, de plus, il fournit la précision de la solution.

V.2. Problème d'optimisation contrainte

Nous présentons ici un exemple très simple d'optimisation contrainte qui met en évidence l'efficacité de la méthode proposée pour vérifier les contraintes et la solution trouvée.

Soit à déterminer :

$$\text{Inf } F(X) = (X_1 - 2)^2 + (X_2 - 1)^2, \tag{17}$$

sous les contraintes :

$$h_1(X) = X_1 - 2X_2 + 1 = 0, \tag{18}$$

$$g_1(X) = \frac{-X_1^2}{4} - X_2^2 + 1 \geqq 0. \tag{19}$$

D'après ce qu'il a été dit précédemment, nous avons transformé la contrainte d'inégalité (19) en contrainte d'égalité (20) définie par :

$$\left.\begin{aligned} g_1(X) = \frac{-X_1^2}{4} - X_2^2 + 1 - X_3 = 0, \\ X_3 \geqq 0. \end{aligned}\right\} \tag{20}$$

Les conditions de Kuhn et Tucker s'expriment par :

$$\left.\begin{aligned} C1 &= 2(X_1+2)+\mu+\frac{\lambda X_1}{2}=0,\\ C2 &= 2(X_2-1)-2\mu+2\lambda X_2=0,\\ C3 &= X_1-2X_2+1=0,\\ C4 &= \frac{-X_1^2}{4}-X_2^2+1-X_3=0,\\ C5 &= \lambda\left(-\frac{X_1^2}{4}-X_2^2+1\right)=0,\\ X_3 &\geqq 0, \qquad \mu \geqq 0. \end{aligned}\right\} \tag{21}$$

Nous avons résolu ce problème en utilisant le logiciel de Permutation-Perturbation sur CDC 7600. Nous avons obtenu trois solutions informatiquement satisfaisantes qui vérifient les contraintes et les conditions de Kuhn et Tucker.

C 1 et C 2 sont vérifiées lorsque leurs nombres de chiffres significatifs NC 1 et NC 2 sont tous deux inférieurs à 1.

C 3 et C 4 étant linéaires, on les vérifie par la méthode des résidus normés exposés dans la première partie. Si ρ^*_{C3} et ρ^*_{C4} sont tous deux inférieurs à 1, alors les contraintes C 3 et C 4 sont satisfaites.

Les résultats obtenus sont présentés dans le Tableau V.

TABLEAU V

	1re résolution	2e résolution	3e résolution
X 1	0.822875655532297	0.822875655532293	0.822875655532297
X 2	0.911437827766147	0.911437827766146	0.911437827766147
X 3	$-1.77635683940026.10^{-14}$	0.	$-1.77635683940025.10^{-14}$
μ	1.59449111825231	1.59449111825232	1.59449111825231
λ	1.84659143960610	1.846591143960610	1.84659143960610
NC 1	0.	0.	0.
NC 2	0.	0.	0.
ρ^*_{C3}	0.	0.	0.
ρ^*_{C4}	0.1	0.1	0.1

Avec les trois valeurs des inconnues, il est aisé de calculer leur nombre de chiffres décimaux significatifs exacts, comme nous l'avons expliqué précédemment.

En appelant respectivement NCX 1, NCX 2, NCX 3, NC μ, NC λ, le nombre de chiffres décimaux significatifs exacts de X 1, X 2, X 3, μ et λ, nous obtenons :

$$\text{NCX } 1 = 14,$$
$$\text{NCX } 2 = 14,$$
$$\text{NCX } 3 = 0,$$
$$\text{NC } \mu = 14,$$
$$\text{NC } \lambda = 15.$$

Nous tombons ici sur le cas particulier où l'un des nombres significatifs, NCX 3, est nul. Ou bien X 3 ne peut pas être calculé car la précision de l'ordinateur est trop faible, ou bien $X\,3 = 0$. Pour lever cette indétermination, il suffit de vérifier la contrainte C 4 avec les valeurs de X 1, X 2, μ, λ trouvées et $X\,3 = 0$.

Pour ces valeurs, on obtient : $\rho^*_{C4} = 0.5$. La contrainte est donc satisfaite. Par conséquent, la solution est :

$$X\,1 = 0.82287565553229, \qquad \text{NCX } 1 = 14,$$
$$X\,2 = 0.91143782776614, \qquad \text{NCX } 2 = 14,$$
$$X\,3 = 0. \quad \text{NCX } 3 = 15, \quad \mu = 1.59449111822523, \quad \text{NC } \mu = 14,$$
$$\lambda = 1.84659143960610, \qquad \text{NC } \lambda = 75.$$

CONCLUSION

Nous avons présenté dans cet article une application de la méthode de Permutation-Perturbation aux algorithmes d'optimisation.

L'utilisation de cette méthode permet, pour les méthodes d'optimisation contraintes ou non contraintes, d'arrêter le processus itératif dès qu'une solution informatique satisfaisante est atteinte et d'estimer la précision de cette solution.

En outre, cette méthode permet de tenir compte des erreurs sur les données du problème.

La méthode de Permutation-Perturbation permet donc de faire faire automatiquement à l'ordinateur l'analyse de la propagation des erreurs dues à l'arithmétique à précision limitée de la machine, ainsi que l'estimation des conséquences des erreurs de données sur les résultats de toute optimisation.

Le logiciel tel qu'il est proposé ici fournit donc pour tout problème d'optimisation le résultat avec sa précision associée. Ce problème n'avait jamais été résolu jusqu'ici.

BIBLIOGRAPHIE

1. J. ABADIE, *The GRG Method for Nonlinear Programming Design and Implementation of Optimization Software. In:* HARVEY J. GREENBERG, éd., Sijthoff & Noordhoff, 1978, p. 335-362.
2. D. M. HIMMELBLAU, *Numerical Methods for Nonlinear Optimization. In:* F. A. LOOTSMA, éd., Academic Press, 1972, p. 69-73.
3. M. LA PORTE et J. VIGNES, *Algorithmes numériques, analyse et mise en œuvre*, tome 1, Technip, ed., Paris, 1974.
4. M. LA PORTE et J. VIGNES, *Étude statistique des erreurs dans l'arithmétique des ordinateurs; application au contrôle des résultats d'algorithmes numériques*, Numer. Math., vol. 23, 1974, p. 63-72.
5. M. MAILLÉ, *Some Methods to Estimate Accuracy of Measures or Numerical Computations. Proceedings of Mathematics for Computer Science*, Colloque international AFCET, Paris, 1982, p. 495-503.
6. P. TOLLA, *Stabilisation et accélération d'algorithmes de Programmation mathématique: Gradient Réduit Généralisé et Moindres Carrés*, Actes du Symposium AFCET « Les Mathématiques pour l'Informatique », Paris, 1982.
7. P. TOLLA, *Linear and Nonlinear Programming Software Validity*. Math. and Comp. in Sim., vol. XXV, 1983, p. 39-42.
8. J. VIGNES et M. LA PORTE, *Error Analysis in Computing*. Proceedings of IFIP Congress, Stockholm, 1974, p. 609-614.
9. J. VIGNES, *Étude et mise en œuvre d'algorithmes de recherche d'un extremum d'une fonction de plusieurs variables*. Thèse d'État, Paris, 1969.
10. J. VIGNES, *New Methods for Evaluating the Validity of the Results of Mathematical Computations*. Math. and Comp. in Sim. Vol. XX, n° 4, 1978, p. 227-249.

```
      FUNCTION PEPER(P,N)
      DIMENSION  P(20)
C     PERTURBATION DES P(I)
      DO 1 I=1,N
      IPILE=HASARD(0.,1.99999999999999)
      IF(IPILE.EQ.0) GO TO 1
      P(I)=XV(P(I),SIGN(FLOAT(IPILE),P(I)))
    1 CONTINUE
      IF(N.LT.3)GO TO 4
C     PERMUTATION DES P(I)
      DO 2 I=1,N
      I2=HASARD(FLOAT(I),FLOAT(N))
      STOCK=P(I)
      P(I)=P(I2)
    2 P(I2)=STOCK
C     CALCUL DE LA SOMME DES P(I) PERTURBES ET PERMUTES
      S=0.
      DO 3 I=1,N
      S=S+P(I)
      I2=HASARD(0.,1.99999999999999)
      IF(I2.EQ.0) GO TO 3
      S=XV(S,SIGN(FLOAT(I2),S))
    3 CONTINUE
      PEPER=S
      RETURN
    4 IF(N.EQ.2)GO TO 5
C     CAS OU N=1
      PEPER=P(N)
      RETURN
C     CAS OU N=2
    5 S=P(1)+P(2)
      I2=HASARD(0.,1.99999999999999)
      IF(I2.EQ.0) GO TO 6
      S=XV(S,SIGN(FLOAT(I2),S))
    6 PEPER=S
      RETURN
      END
  FUNCTION XV(X,S)
  PRE(Z)=OR(00004000000000000000B,AND(ABS(Z),77770000000000000000B))
  DER(Z)=OR(00000000000000000001B,AND(ABS(Z),77770000000000000000B))
  IF(X.EQ.0.)GO TO 1
  XV=X+SIGN(DER(X),S)
  IF(PRE(X).GT.PRE(XV))XV=XV-SIGN(DER(XV),S)
  RETURN
1 XV=SIGN(PRE(0.),S)
  RETURN
  END
```

Remarque : La fonction PEPER fait appel à une fonction Hasard qui génère aléatoirement les valeurs 0 et 1.

ANNEXES

Texte du sous-programme FORTRAN de résolution d'un système d'équations non linéaire par la méthode d'optimisation avec test d'arrêt optimal et calcul de la précision de la solution.

```
      SUBROUTINE RSNLX2(FONDER,X,F,NINC,NBITER,IMP,IRES,BLOC)
      DIMENSION X(1),F(1),BLOC(1),IMP(4)
C        FONDER NOM DU SOUS-PROGRAMME QUI POUR TOUT VECTEUR X CALCULE
C               LES FONTIONS F ET LEURS DERIVEES PREMIERES DF.
C               LE NOM DE CE SOUS-PROGRAMME DOIT APPARAITRE DANS UN
C               ORDRE EXTERNAL DU PROGRAMME APPELANT.
C               LE CALCUL DES FONCTIONS F ET DE LEURS DERIVEES DF SE
C               FAIT DANS CE SOUS-PROGRAMME A L AIDE DE LA FUNCTION
C               PEPER(P,N)
C                       P EST LE VECTEUR DES MONOMES DE F OU DE DF
C                       N EST LE NOMBRE DE MONOMES DE F OU DE DF
C        X          BLOC DES INCONNUES
C        F          BLOC DES FONCTIONS F(I) CORRESPONDANT A CHAQUE EQUATION
C        NINC       NOMBRE D INCONNUES
C        NBITER     NOMBRE MAXIMUM D ITERATIONS
C        IMP(1))    =1  IMPRESSION DES CONDITIONS INITIALES
C                   =O  SINON
C        IMP(2)     =1  IMPRESSION DU RESULTAT FINAL
C                   =O  SINON
C        IMP(3)     PAS DE L IMPRESSION DES RESULTATS INTERMEDIAIRES (LES CONDI
C                   TIONS INITIALES ET LE RESULTAT FINAL NON COMPRIS)
C        IMP(4)     =1  IMPRESSION INTERMEDIAIRES DETAILLEES (IL FAUDRA IMP(3)=1)
C                   =O  SINON
C        BLOC       BLOC DE TRAVAIL DIMENSIONNE AU MINIMUM A
C                   NINC*(6+NINC+NC)+SUP(NC,NINC)
C        IRES       =O  SI TOUT S EST BIEN PASSE
C        IRES       =1  SI LA MATRICE DU SYSTEME LINEAIRE EST DEGENEREE (MAIS UN
C                       ESSAI A ETE TENTE DANS LA DIRECTION DU GRADIENT)
C        IRES       =2  SI LA PRECISION DEMANDEE N A PAS ETE OBTENUE EN NBITER
C                       ITERATIONS
C        IRES       =3  SI ON NE PEUT PLUS AMELIORER LA SOLUTION OBTENUE BIEN QUE
C                   LA PRECISION DEMANDEE N AIT PAS ETE ATTEINTE
      EXTERNAL FONDER
      IXR=1
      ICA=IXR+NINC
      IDX=ICA+NINC*(NINC+1)
      IXRR=IDX+MAXO(NINC,NINC)
      IFN=IXRR+NINC
      IFNR=IFN+NINC
      IFNBAS=IFNR+NINC
      IDP=IFNBAS+NINC
      CALL RSNLX3(FONDER,X,F,NINC,NBITER,IMP,IRES,BLOC(IXR),BLOC(ICA),BL
     1OC(IDX),BLOC(IXRR),BLOC(IFN),BLOC(IFNR),BLOC(IFNBAS),BLOC(IDP))
      RETURN
      END
```

```
      SUBROUTINE RSNLX3(FONDER,X,F,NINC,NBITER,IMP,IRES,XR,CA,DX,XRR,FN,
     1FNR,FNBAS,DP)
      DIMENSION X(1),F(1),          IMP(5)
      DIMENSION XR(1),CA(NINC,1),DX(1),XRR(1),FN(1),FNR(1),FNBAS(1),DP(N
     1INC,1)
      DIMENSION FSTOK(20,3)
      EXTERNAL FONDER
      DIMENSION DF2(10),FPP(10,5),FM(10),VAR(10),ERM(10),C(10)
      DIMENSION XCRESM(10),XCRES(10,5) ,XCVAR(10),ERMXC(10),CXC(10)
      DIMENSION CXCRES(10,5),XO(10)
  100 FORMAT(1H1)
  101 FORMAT(////)
  102 FORMAT(5X,*SOLUTION OBTENUE PAR LA DIRECTION DU SYSTEME LINEAIRE*/
     1//)
  103 FORMAT(5X,*SOLUTION OBTENUE PAR LA DIRECTION DU GRADIENT*///)
  104 FORMAT(5X,*SOLUTION OBTENUE PAR LA DIRECTION DE LA BISSECTRICE*///
     1)
  105 FORMAT(//30X,*SYSTEME LINEAIRE*)
  106 FORMAT(30X,*EQ=*,E22.15)
  107 FORMAT(/,30X,*SOLUTION DANS LA DIRECTION DONNEE PAR LE SYSTEME LIN
     1EAIRE*)
  108 FORMAT(30X,2HX(,I2,2H)=,E22.15)
  109 FORMAT(//30X,*BISSECTRICE DES DEUX PRECEDENTES DIRECTIONS DU GRADI
     1ENT*)
  110 FORMAT(//30X,*GRADIENT APRES SUCCES SYSTEME LINEAIRE*)
  111 FORMAT(//30X,*BISSECTRICE APRES SUCCES SYSTEME LINEAIRE*)
  112 FORMAT(/30X,*SOLUTION DANS LA DIRECTION DU GRADIENT*)
  113 FORMAT(/30X,*SOLUTION DANS LA DIRECTION DE LA BISSECTRICE*)
  114 FORMAT(//30X,*GRADIENT APRES ECHEC SYSTEME LINEAIRE*)
  115 FORMAT(//30X,*BISSECTRICE APRES ECHEC SYSTEME LINEAIRE*)
  116 FORMAT(/30X,*SOLUTION DANS LA DIRECTION DU GRADIENT*)
  117 FORMAT(/30X,*SOLUTION DANS LA DIRECTION DE LA BISSECTRICE*)
  118 FORMAT(1H ,///     50X,17HMATRICE DEGENEREE,///)
  119 FORMAT(1H ,///       50X,34HPRECISION DEMANDEE NON OBTENUE EN ,I3,
     112H ITERATIONS.///)
  120 FORMAT(///40X,*LA SOLUTION OBTENUE A L ITERATION*,I3,* NE PEUT PLU
     1S ETRE AMELIOREE*)
  121 FORMAT(5X,2HX(,I2,4H) = ,E22.15)
  122 FORMAT(/5X,3HEQ(,I3,4H) = ,E22.15)
      NC=NINC
      ICOMPT=0
      JCOMPT=0
      DO 70 I=1,NINC
   70 XO(I)=X(I)
   71 JJ=0
      DO 72 I=1,NINC
   72 X(I)=XO(I)
      IRES=0
      NM1=NINC-1
      NN=NINC+1
      N=0
      IF(IMP(3).EQ.0)IMP(3)=NBITER+2
      CALL FONDER(X,NINC,    F,DP,3)
      EQ=0.
      DO   1 I=1,NC
    1 EQ=EQ+F(I)**2
      EQBAS=EQ
      IMP1=IMP(1)+IMP(2)+IMP(3)+IMP(4)
      IF(IMP1.NE.0)PRINT 100
      IF(IMP(1).EQ.0)GO TO 8
      PRINT 121,(I,X(I),I=1,NINC)
      PRINT 122,N,EQBAS
      PRINT 101
      GO TO 8
```

```
C       TEST D ARRET OPTIMAL
      2 JJ=JJ+1
        CALL FONDER(X,NINC,   F,DP,3)
        DO 1000 I=1,NINC
        DF2(I)=0.
        DO 1000 K=1,NINC
 1000 DF2(I)=DF2(I)+2.*F(K)*DP(K,I)
        DO 531 I=1,NINC
        FPP(I,JJ)=DF2(I)
  531 CONTINUE
        IF(JJ.LT.3)GO TO 2
C       CALCUL DES C(F) DES DF2(I)
C       CALCUL DES DF2(F) MOYENS
        DO 534 I=1,NINC
        FM(I)=0.
        DO 534 K=1,JJ
  534 FM(I)=FM(I)+FPP(I,K)
        DO 535 I=1,NINC
  535 FM(I)=FM(I)/FLOAT(JJ)
C       CALCUL DES VARIANCES DES DF2(I)
        DO 536 I=1,NINC
        VAR(I)=0.
        DO 536 K=1,JJ
  536 VAR(I)=VAR(I)+(FPP(I,K)-FM(I))**2
        DO 537 I=1,NINC
  537 VAR(I)=VAR(I)/FLOAT(JJ-1)
C       CALCUL DE L ERREUR MOYENNE ERM(I)
        DO 538 I=1,NINC
  538 ERM(I)=SQRT((FPP(I,1)-FM(I))**2+VAR(I))
C       CALCUL DU C(I) DES DF2(I)
        DO 539 I=1,NINC
        IF(ERM(I).EQ.0.)GO TO 540
        G=ABS(FPP(I,1)/ERM(I))
        IF(G.EQ.0.)GO TO 541
        C(I)=ALOG10(G)
        GO TO 539
  541 C(I)=0.
        GO TO 539
  540 C(I)=15.
        IF(VAR(I).EQ.0.)C(I)=0.
  539 CONTINUE
        IF(IT1.EQ.50) GO TO 733
        DO 507 I=1,NINC
        IF(C(I).GE.1.)GO TO 4
  507 CONTINUE
C       CALCUL DU NOMBRE DE CHIFFRES SIGNIFICATIFS DE LA SOLUTION
C       STOCKAGE DU VECTEUR-SOLUTION
  733 ICOMPT=ICOMPT+1
        IF(IMP(3).EQ.0)GO TO 334
        PRINT 199,ICOMPT
  199 FORMAT(1H ,*SOLUTION TROUVEE A LA*,I2,* IEME RESOLUTION*)
        DO 333 K=1,NINC
  333 PRINT 123,K,X(K)
  123 FORMAT(21X,2HX(,I2,2H)=,E22.15)
        PRINT 200,EQBAS
  200 FORMAT(21X,*EQ=*,E22.15)
  334 DO 550 I=1,NINC
  550 XCRES(I,ICOMPT)=X(I)
        IF(ICOMPT.LT.2)GO TO 71
C       CALCUL DES X(I) MOYENS
        DO 551 I=1,NINC
        XCRESM(I)=0.
        DO 551 K=1,ICOMPT
  551 XCRESM(I)=XCRESM(I)+XCRES(I,K)
```

```
      DO 552 I=1,NINC
  552 XCRESM(I)=XCRESM(I)/FLOAT(ICOMPT)
C     CALCUL DE LA VARIANCE DU VECTEUR SOLUTION
      DO 553 I=1,NINC
      XCVAR(I)=0.
      DO 553 K=1,ICOMPT
  553 XCVAR(I)=XCVAR(I)+(XCRES(I,K)-XCRESM(I))**2
      DO 554 I=1,NINC
  554 XCVAR(I)=XCVAR(I)/FLOAT(ICOMPT-1)
C     CALCUL DE L ERREUR MOYENNE SUR LE VECTEUR SOLUTION
      DO 555 I=1,NINC
  555 ERMXC(I)=SQRT((XCRES(I,1)-XCRESM(I))**2+XCVAR(I))
C     CALCUL DES CXC(I)
      DO 556 I=1,NINC
      IF(ERMXC(I).EQ.0.)GO TO 557
      G2=ABS(XCRES(I,1)/ERMXC(I))
      IF(G2.EQ.0.)GO TO 558
      CXC(I)=ALOG10(G2)
      GO TO 556
  557 CXC(I)=15.
      IF(XCVAR(I).EQ.0.)CXC(I)=0.
      GO TO 556
  558 CXC(I)=0.
  556 CONTINUE
C     MISE A ZERO DES XC(I) DONT LE CXC(I) EST INFERIEUR A 1
      DO 570 I=1,NINC
      IF(CXC(I).LT.1.)GO TO 571
      GO TO 570
  571 X(I)=0.
      CXC(I)=15.
  570 CONTINUE
C     STOCKAGE DES CXC(I)
      JCOMPT=JCOMPT+1
      DO 559 I=1,NINC
  559 CXCRES(I,JCOMPT)=CXC(I)
      IF(JCOMPT.GE.5)GO TO 5000
      IF(JCOMPT.EQ.1)GO TO 71
      DO 561 I=1,NINC
      IF(CXCRES(I,JCOMPT)-CXCRES(I,JCOMPT-1).GT.0.5)GO TO 71
  561 CONTINUE
C     IMPRESSION DE LA PRECISION DU VECTEUR SOLUTION
      PRINT 124
  124 FORMAT(1HO,4X,*SOLUTION FINALE TROUVEE*)
      DO 509 K=1,NINC
      IC=CXCRES(K,JCOMPT)+0.5
  509 PRINT 125,K,IC+7,IC,X(K),20-IC,IC
  125 FORMAT(6X,*X(*,I3,*)=*,E=.=,=X,*C=*,I3)
      DO 900 L=1,3
      CALL FONDER(X,NINC,F,DP,1)
      DO 900 I=1,NINC
  900 FSTOK(I,L)=F(I)
C
      DO 901 I=1,NINC
      FM(I)=0.
      DO 901 L=1,3
  901 FM(I)=FM(I)+FSTOK(I,L)
      DO 902 I=1,NINC
  902 FM(I)=FM(I)/3.
C
      DO 903 I=1,NINC
      VAR(I)=0.
      DO 903 L=1,3
  903 VAR(I)=VAR(I)+(FSTOK(I,L)-FM(I))**2
```

```
      DO 904 I=1,NINC
  904 VAR(I)=VAR(I)/2.
C
      DO 905 I=1,NINC
  905 ERM(I)=SQRT((FSTOK(I,1)-FM(I))**2+VAR(I))
C
      DO 939 I=1,NINC
      IF(ERM(I).EQ.0.)GO TO 940
      G=ABS(FSTOK(I,1))/ERM(I)
      IF(G.EQ.0.)GO TO 941
      C(I)=ALOG10(G)
      GO TO 939
  941 C(I)=0.
      GO TO 939
  940 C(I)=15.
      IF(VAR(I).EQ.0.)C(I)=0.
  939 CONTINUE
      DO 942 I=1,NINC
      IF(C(I).GT.1.)GO TO 951
  942 CONTINUE
      PRINT 8500
 8500 FORMAT(6X,*LA SOLUTION TROUVEE EST SOLUTION DU SYSTEME NON LINEAIR
     1E*)
      DO 952 I=1,NINC
  952 F(I)=0.
      PRINT 8503 ,(I,F(I),I=1,NINC)
 8503  FORMAT(40X,*F(*,I3,*)=*,F3.0)
      RETURN
  951 PRINT 8501
 8501 FORMAT(6X,*LA SOLUTION TROUVEE N EST PAS SOLUTION DU SYSTEME NON L
     1INEAIRE CONSIDERE*)
      DO 953 I=1,NINC
      IC=C(I)+0.5
  953 PRINT 8502,I,IC+7,IC,F(I),20-IC,IC
 8502  FORMAT(40X,*F(*,I3,*)=*,E=.=,=X,*C=*,I3)
      RETURN
 5000 PRINT 126
  126 FORMAT(1HO,4X,*PRECISION DE LA SOLUTION NON TROUVEE APRES 5 RUNS*)
      RETURN
    4 JJ=0
      DO  5 I=1,NC
    5 F(I)=DX(I)
      CALL FONDER(X,NINC,   F,DP,2)
      IF(MOD(N,IMP(3)).NE.0)GO TO 8
      PRINT 121,(I,X(I),I=1,NINC)
      PRINT 122,N,EQBAS
      IF(IPASS.EQ.1)GO TO 6
      IF(IPASS.EQ.2)GO TO 7
      PRINT 104
      GO TO 8
    6 PRINT 102
      GO TO 8
    7 PRINT 103
C
    8 N=N+1
      Y1=EQBAS
      IBI=0
      IF(N.GT.NBITER) GO TO 64
C
C     RANGEMENT DES FONCTIONS DANS LA MATRICE DU SYSTEME LINEAIRE.
C
      DO 10 I=1,NINC
      XR(I)=X(I)
      CA(I,NN)=0.
      DO  9 J=1,NC
    9 CA(I,NN)=CA(I,NN)-F(J)*DP(J,I)
   10 FN(I)=-CA(I,NN)
```

```
      DO 11 I=1,NINC
      CA(I,I)=0.
      DO 11J=1,NC
   11 CA(I,I)=CA(I,I)+DP(J,I)**2
      NZ=1
      DO 13 I=1,NM1
      NZ=NZ+1
      DO 13 K=NZ,NINC
      CA(I,K)=0.
      DO 12 J=1,NC
   12 CA(I,K)=CA(I,K)+DP(J,I)*DP(J,K)
   13 CA(K,I)=CA(I,K)
C
C     RESOLUTION DU SYSTEME LINEAIRE .
C
      CALL RSL(CA,NINC,NINC,IRES)
      IF(IRES.NE.1)GO TO 14
      PRINT 118
      GO TO 22
   14 DO 15 I=1,NINC
   15 DX(I)=CA(I,NN)
      IF(IMP(4).EQ.1)PRINT 105
      IT1=0
      IT2=0
      ALF=1.
C
C     RECHERCHE DANS LA DIRECTION DONNEE PAR LE SYSTEME LINEAIRE .
C
   16 DO 17 I=1,NINC
   17 X(I)=XR(I)+ALF*DX(I)
      CALL FONDER(X,NINC,   F,DP,1)
      EQ=0.
      DO 18 I=1,NC
   18 EQ=EQ+F(I)**2
      IF(IMP(4).EQ.1)PRINT 106,EQ
      IF(EQ.GE.EQBAS) GO TO 21
      DO 19 I=1,NINC
   19 XRR(I)=X(I)
      IPASS=1
      DO 20 I=1,NC
   20 DX(I)=F(I)
      EQBAS=EQ
      IF(IMP(4).EQ.0)GO TO 23
      PRINT 107
      PRINT 108,(I,XRR(I),I=1,NINC)
      PRINT 106,EQ
      GO TO 23
   21 IT1=IT1+1
      IF(IT1.EQ.50) GO TO 22
      ALF=ALF/2.
      GO TO 16
   22 IT2=1
C
C     DETERMINATION DE LA DIRECTION DE PLUS GRANDE PENTE .
C
   23 DEN=0.
      DO 24 I=1,NINC
      FN(I)=2.*FN(I)
   24 DEN=DEN+FN(I)**2
      DEN=SQRT(DEN)
      DO 25 I=1,NINC
      FN(I)=-FN(I)/DEN
   25 FNR(I)=FN(I)
      IF(N.EQ.1) GO TO 29
```

```
C
C     CALCUL DU COSINUS DE L"ANGLE DE DEUX DIRECTIONS SUCCESSIVES
C     DE PLUS GRANDE PENTE
C
      COTETA=0.
      DO 26 I=1,NINC
   26 COTETA=COTETA+FN(I)*FNBAS(I)
      IF(COTETA.GT.0.) GO TO 29
      IBI=1
      IF(IMP(4).EQ.1)PRINT 109
C
C     CALCUL DE LA BISSECTRICE DES DEUX PRECEDENTES DIRECTIONS
C
      DEN=0.
      DO 27 I=1,NINC
      FN(I)=FN(I)+FNBAS(I)
   27 DEN=DEN+FN(I)**2
      DEN=SQRT(DEN)
      DO 28 I=1,NINC
   28 FN(I)=FN(I)/DEN
   29 IF(IT2.EQ.1) GO TO 47
C
C     RECHERCHE DANS LA DIRECTION DU GRADIENT (OU BISSECTRICE)
C     APRES SUCCES DANS LA DIRECTION DONNEE PAR LE SYSTEME LINEAIRE
C
      IT1=0
      IF(IMP(4).EQ.0)GO TO 30
      IF(IBI.EQ.0)PRINT 110
      IF(IBI.EQ.1)PRINT 111
   30 S=0.
      DO 31 I=1,NINC
      S=S+(X(I)-XR(I))**2
   31 FNBAS(I)=FNR(I)
      ALF=SQRT(S)
   32 DO 33 I=1,NINC
   33 X(I)=XR(I)+ALF*FN(I)
      CALL FONDER(X,NINC,   F,DP,1)
      EQ=0.
      DO 34 I=1,NC
   34 EQ=EQ+F(I)**2
      IF(IMP(4).EQ.1)PRINT 106,EQ
      IF(EQ.GT.EQBAS)GO TO 38
      IF(EQ.LT.EQBAS)GO TO 35
      IF(IT1.EQ.0)GO TO 38
   35 DO 36 I=1,NINC
   36 XRR(I)=X(I)
      DO 37 I=1,NC
   37 DX(I)=F(I)
      EQBAS=EQ
      IF(IT1.NE.0)GO TO 40
      ALF=ALF*2.
      GO TO 32
   38 ALF=ALF/2.
      Y3=EQ
      IT1=IT1+1
      IF(IT1.LE.5)GO TO 32
      DO 39 I=1,NINC
   39 X(I)=XRR(I)
      GO TO 2
   40 IPASS=2+IBI
      ALF=ALF/2.*(3.*Y1-4.*EQ+Y3)/(Y1-2.*EQ+Y3)
      DO 41 I=1,NINC
   41 X(I)=XR(I)+ALF*FN(I)
      CALL FONDER(X,NINC,   F,DP,1)
```

```
      EQ=O.
      DO 42 I=1,NC
   42 EQ=EQ+F(I)**2
      IF(EQ.GT.EQBAS)GO TO 44
      EQBAS=EQ
      DO 43 I=1,NC
   43 DX(I)=F(I)
      GO TO 46
   44 DO 45 I=1,NINC
   45 X(I)=XRR(I)
   46 IF(IMP(4).EQ.O)GO TO 2
      IF(IBI.EQ.O)PRINT 112
      IF(IBI.EQ.1)PRINT 113
      PRINT 108,(I,X(I),I=1,NINC)
      PRINT 106,EQBAS
      GO TO 2
C
C     RECHERCHE DANS LA DIRECTION DU GRADIENT (OU BISSECTRICE) APRES ECHEC  DANS
C      LA DIRECTION DONNEE PAR LE SYSTEME LINEAIRE
C
   47 IT1=O
      IF(IMP(4).EQ.O)GO TO 48
      IF(IBI.EQ.O)PRINT 114
      IF(IBI.EQ.1)PRINT 115
   48 DO 49 I=1,NINC
   49 FNBAS(I)=FNR(I)
      ALF=1.
   50 DO 51 I=1,NINC
   51 X(I)=XR(I)+ALF*FN(I)
      CALL FONDER(X,NINC,   F,DP,1)
      EQ=O.
      DO 52 I=1,NC
   52 EQ=EQ+F(I)**2
      IF(IMP(4).EQ.1)PRINT 106,EQ
      IF(EQ.GT.EQBAS)GO TO 56
      IF(EQ.LT.EQBAS)GO TO 53
      IF(IT1.EQ.O)GO TO 56
   53 DO 54 I=1,NINC
   54 XRR(I)=X(I)
      DO 55 I=1,NC
   55 DX(I)=F(I)
      EQBAS=EQ
      IF(IT1.NE.O)GO TO 57
      ALF=ALF*2.
      GO TO 50
   56 IT1=IT1+1
      Y3=EQ
      IF(IT1.EQ.50) GO TO 65
      ALF=ALF/2.
      GO TO 50
   57 IPASS=2+IBI
      ALF=ALF/2.*(3.*Y1-4.*EQ+Y3)/(Y1-2.*EQ+Y3)
      DO 58 I=1,NINC
   58 X(I)=XR(I)+ALF*FN(I)
      CALL FONDER(X,NINC,   F,DP,1)
```

```
      EQ=0.
      DO 59 I=1,NC
   59 EQ=EQ+F(I)**2
      IF(EQ.GT.EQBAS)GO TO 61
      EQBAS=EQ
      DO 60 I=1,NC
   60 DX(I)=F(I)
      GO TO 63
   61 DO 62 I=1,NINC
   62 X(I)=XRR(I)
   63 IF(IMP(4).EQ.0)GO TO 2
      IF(IBI.EQ.0)PRINT 116
      IF(IBI.EQ.1)PRINT 117
      PRINT 108,(I,X(I),I=1,NINC)
      PRINT 106,EQBAS
      GO TO 2
   64 PRINT 119,NBITER
      IRES=2
      GO TO 67
   65 N=N-1
      IRES=3
      DO 66 I=1,NINC
   66 X(I)=XR(I)
      PRINT 120,N
   67 CONTINUE
C
      IF(IMP(2).EQ.0) GO TO 2
      PRINT 121,(I,X(I),I=1,NINC)
      PRINT 122,N,EQBAS
      GO TO 2
      END
```

Remarque : Le sous-programme RSNLX 3 fait appel à un sous-programme RSL de résolution de systèmes d'équations linéaires.

Exemple d'utilisation du sous-programme RSNLX 2.

Considérons le système d'ordre 2 suivant :

$$\begin{cases} f_1 = 7\,x_1^2 + 3\,x_1\,x_2 + 4\,x_1 - x_2 - 41 = 0, \\ f_2 = 10\,x_1^2 + 4\,x_1\,x_2 + 5\,x_1 - 2\,x_2 - 56 = 0, \end{cases}$$

qui admet comme solution $x_1^* = 2$, $x_2^* = 1$.

Pour résoudre sur ordinateur ce système d'équations non linéaires par la méthode d'optimisation décrite précédemment avec test d'arrêt optimal et détermination de la précision sur la solution, on doit exprimer chacune des équations du système non linéaire et leurs dérivées partielles sous forme d'une suite de monômes dans un sous-programme. Les permutations-perturbations sont assurées par la fonction PEPER décrite précédemment.

(*a*) *Programme principal*

```
      PROGRAM TEST(INPUT,OUTPUT,TAPE5=INPUT)
      EXTERNAL FONDER
      DIMENSION F(2),X(2),IMP(4),BLOC(22)
      NINC=2
      NBITER=50
      IMP(1)=IMP(2)=1
      IMP(3)=IMP(4)=0
    3 READ(5,100)X(1),X(2)
  100 FORMAT(2E7.3)
      IF(EOF(5))1,2
    2 CALL RSNLX2(FONDER,X,F,NINC,NBITER,IMP,IRES,BLOC)
      GO TO 3
    1 STOP
      END
```

(*b*) *Sous-programme FONDER*

Dans ce sous-programme sont exprimées, sous forme de suite de monômes, d'une part chacune des équations du système et d'autre part chacune des dérivées partielles des fonctions.

Notons que l'index ID nécessairement présent comme argument du sous-programme permet de faire calculer par ce sous-programme, soit seulement les valeurs des fonctions ou des dérivées, soit l'ensemble des valeurs des fonctions et des dérivées.

```
      SUBROUTINE FONDER(X,NINC,F,DP,ID)
      DIMENSION  X(1),F(1),DP(NINC,1),P(5)
      IF(ID.EQ.2) GO TO 10
      P(1)=7.*X(1)**2
      P(2)=3.*X(1)*X(2)
      P(3)=4.*X(1)
      P(4)=-X(2)-41.
      F(1)=PEPER(P,4)
      P(1)=10.*X(1)**2
      P(2)=4.*X(1)*X(2)
      P(3)=5.*X(1)
      P(4)=-2.*X(2)-56.
      F(2)=PEPER(P,4)
      IF(ID.EQ.1) RETURN
10    P(1)=14.*X(1)
      P(2)=3.*X(2)+4.
      DP(1,1)=PEPER(P,2)
      P(1)=3.*X(1)-1.
      DP(1,2)=PEPER(P,1)
      P(1)=20.*X(1)
      P(2)=4.*X(2)+5.
      DP(2,1)=PEPER(P,2)
      P(1)=4.*X(1)-2.
      DP(2,2)=PEPER(P,1)
      RETURN
      END
```

(*c*) *Résultats obtenus sur CDC* 7600

1. $x_1^0 = -1 \quad x_2^0 = 50$

```
X( 1) =  -.100000000000000E+01
X( 2) =   .500000000000000E+02

EQ(  0) =   .179845000000002E+06


SOLUTION TROUVEE A LA 1 IEME RESOLUTION
                    X( 1)=  .199999999999999E+01
                    X( 2)=  .999999999999950E+00
                    EQ=  .233937016987653E-23

X( 1) =  -.100000000000000E+01
X( 2) =   .500000000000000E+02

EQ(  0) =   .179845000000001E+06

SOLUTION TROUVEE A LA 2 IEME RESOLUTION
                    X( 1)=  .200000000000009E+01
                    X( 2)=  .999999999999346E+00
                    EQ=  .516987882845642E-25
```

```
X( 1) =  -.100000000000000E+01
X( 2) =   .500000000000000E+02

EQ(  0) =    .179845000000000E+06



SOLUTION TROUVEE A LA 3 IEME RESOLUTION
                    X( 1)=  .200000000000001E+01
                    X( 2)=  .999999999999758E+00
                    EQ=   .103397576569128E-23

    SOLUTION FINALE TROUVEE
     X(  1)=  .200000000000000E+01       C= 14
     X(  2)=  .100000000000E+01          C= 12
     LA SOLUTION TROUVEE EST SOLUTION DU SYSTEME NON LINEAIRE
                                         F(  1)= 0.
                                         F(  2)= 0.
```

2. $x_1^0 = -5 \quad x_2^0 = 22$

```
X( 1) =  -.500000000000000E+01
X( 2) =   .220000000000000E+02

EQ(  0) =    .155869000000001E+06


X( 1)=  -.202538615252618E+01
X( 2)=  -.261552421503006E+01

EQ( 20)=  .525954133862228E+01

SOLUTION TROUVEE A LA 1 IEME RESOLUTION
                    X( 1)= -.202538615252618E+01
                    X( 2)= -.261552421503006E+01
                    EQ=   .525954133862228E+01
```

```
X( 1) =  -.500000000000000E+01
X( 2) =   .220000000000000E+02

EQ(  0) =   .155868999999997E+06



X( 1)=  -.202538566416861E+01
X( 2)=  -.261552641352449E+01

EQ( 21)=  .525954133862223E+01
SOLUTION TROUVEE A LA 2 IEME RESOLUTION
                    X( 1)= -.202538566416861E+01
                    X( 2)= -.261552641352449E+01
                    EQ=   .525954133862223E+01



X( 1) =  -.500000000000000E+01
X( 2) =   .220000000000000E+02

EQ(  0) =   .155868999999998E+06



X( 1)=  -.202538584795113E+01
X( 2)=  -.261552555939312E+01

EQ( 21)=  .525954133862186E+01
SOLUTION TROUVEE A LA 3 IEME RESOLUTION
                    X( 1)= -.202538584795113E+01
                    X( 2)= -.261552555939312E+01
                    EQ=   .525954133862186E+01

    SOLUTION FINALE TROUVEE
     X(  1)= -.2025386E+01              C=  7
     X(  2)= -.261553E+01               C=  6
     LA SOLUTION TROUVEE N EST PAS SOLUTION DU SYSTEME NON LINEAIRE CONSIDERE
                                        F(  1)= -.1878357641998E+01      C= 13
                                        F(  2)=  .1315794021635E+01      C= 13
```