RAIRO. RECHERCHE OPÉRATIONNELLE

J. VIGNES

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

*RAIRO. Recherche opérationnelle*, tome 18, nº 1 (1984), p. 1-18

<http://www.numdam.org/item?id=RO_1984__18_1_1_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 18, n° 1, février 1984, p. 1 à 18)



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

par J. VIGNES (1)

Résumé. — *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 dans ces deux articles des méthodes orginales permettant de résoudre ces problèmes. Ce premier article traite des méthodes d'implémentation. Les logiciels correspondants sont présentés dans le prochain numéro de la Revue.*

Mots clés : Optimisation, analyse des erreurs d'arrondi, critère d'arrêt optimal, précision de résultats d'algorithmes, validité des logiciels numériques.

Abstract. — *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 computed solution approximates the mathematical solution, and to what degree of accuracy.*

*New methods of solving these problems are presented in this first part, and the corresponding software in the second part, to appear in the next issue.*

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

(*) 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, 0399-0559/1984/1/$ 5.00

# Première partie

## ASPECT MÉTHODOLOGIQUE

### 1. INTRODUCTION

Les algorithmes d'optimisation sont classés en deux catégories : ceux qui permettent de résoudre des problèmes d'optimisation sans contraintes et ceux relatifs aux optimisations avec contraintes.

Du point de vue mathématique :

– les algorithmes d'optimisation sans contrainte consistent à résoudre :

$$\inf_{\sup} \mathscr{F}(\mathscr{X}), \tag{1}$$

$\mathscr{F}$ étant une fonction de $\mathbb{R}^n \to \mathbb{R}$, $\mathscr{F}(\mathscr{X}) \in \mathbb{R}$, $\mathscr{X} \in \mathbb{R}^n$;

– les algorithmes d'optimisation avec contrainte consistent à résoudre :

$$\inf_{\sup} \mathscr{F}(\mathscr{X}),$$

avec les conditions supplémentaires suivantes :

$$\left.\begin{array}{c} h_i(\mathscr{X}) = 0, \quad i = 1, \ldots, m, \quad m \in \mathbb{N}, \\ h_i \text{ sont les fonctions de } \mathbb{R}^n \to \mathbb{R}, \\ g_j(\mathscr{X}) \geqq 0, \quad j = m+1, \ldots, p, \quad p \in \mathbb{N}, \\ g_j \text{ sont les fonctions de } \mathbb{R}^n \to \mathbb{R}. \end{array}\right\} \tag{2}$$

Tous les algorithmes d'optimisation sont itératifs.

Du point de vue de l'informatique, c'est-à-dire de leur implémentation sur ordinateur, trois problèmes se posent :

– comment arrêter le processus itératif ?

– comment contrôler la validité des résultats fournis par l'ordinateur ?

– quelle est la précision de ces résultats compte tenu du fait que les calculs sont effectués avec une arithmétique à précision limitée (virgule flottante) ?

Nous présentons ici des méthodes originales permettant de résoudre ces problèmes.

## 2. LE CONCEPT DE TEST D'ARRÊT OPTIMAL

### 2.1. Optimisation non contrainte

Lorsque l'on implémente sur ordinateur un algorithme quelconque permettant de résoudre $\inf_{\sup} \mathcal{F}(\mathcal{X})$, c'est en fait le problème $\inf_{\sup} F(X)$ que l'on va résoudre, $F$ étant une représentation informatique de $\mathcal{F}$, $F(X) \in \mathbb{F}$ et $X$ étant celle de $\mathcal{X}$, $X \in \mathbb{F}^n$, $\mathbb{F}$ étant l'ensemble des représentations des valeurs numériques mises sous forme virgule flottante normalisée qui peuvent être représentées sur l'ordinateur considéré.

Les tests d'arrêt classiques généralement utilisés consistent à arrêter le processus itératif lorsque :

$$\| v \| \leqq \varepsilon, \tag{3}$$

$v$ étant soit la différence entre deux itérations successives $q$ et $q+1$ d'une fonctionnelle, soit la valeur à l'itération $q$ de cette fonctionnelle :

$$\left.\begin{array}{c} F(X^{(q+1)}) - F(X^{(q)}) \quad \text{ou} \quad F(X^{(q)}), \\ X^{(q+1)} - X^{(q)}, \\ \operatorname{grad} F(X^{(q+1)}) - \operatorname{grad} F(X^{(q)}) \quad \text{ou} \quad \operatorname{grad} F(X^{(q)}), \end{array}\right\} \tag{4}$$

$\operatorname{grad} F(X)$ étant le gradient de $F(X)$. $\varepsilon \in \mathbb{R}^+$ étant choisi arbitrairement.

Ces tests d'arrêt ne sont pas satisfaisants, car :

– si $\varepsilon$ est choisi trop grand, le processus itératif est arrêté avant d'avoir obtenu la meilleure solution que l'ordinateur peut fournir;

– si $\varepsilon$ est choisi trop petit, un grand nombre d'itérations sont alors faites sans pour autant améliorer la précision réelle de la solution. Ces itérations sont par conséquent absolument inutiles.

Cet inconvénient majeur sur le choix d'$\varepsilon$ est bien connu et a été souligné dans [2], aussi des tests d'arrêt portant sur des combinaisons des précédents ou des variantes telles celles proposées dans [1] sont aussi utilisés. Mais tous ces tests d'arrêt nécessitent le choix d'$\varepsilon$ et, par conséquent, ne sont pas satisfaisants.

Le concept de test d'arrêt optimal repose sur le fait qu'à la solution une fonctionnelle doit être satisfaite.

Soit $\mathcal{X}^* \in \mathbb{R}$ l'optimum (maximum ou minimum) de $\mathcal{F}(\mathcal{X})$. Au point $\mathcal{X}^*$ on a :

$$\operatorname{grad} \mathcal{F}(\mathcal{X}^*) = 0 \quad \text{ou} \quad \| \operatorname{grad} \mathcal{F}(\mathcal{X}^*) \| = 0. \tag{5}$$

Ainsi, du point de vue mathématique, le processus itératif doit être interrompu dès que ces fonctionnelles sont satisfaites.

Certes, lorsque $\mathscr{F}(\mathscr{X})$ présente un col, en ce point le gradient de $\mathscr{F}$ est aussi nul; or, compte tenu des propriétés mathématiques d'un col, il est aisé de voir que l'optimum n'est pas atteint.

Mais, du point de vue de l'informatique, même si nous faisons calculer grad $F(X^*)$ ou $\|\text{grad } F(X^*)\|$, $X^* \in \mathbb{F}$ étant la représentation machine de $\mathscr{X}^* \in \mathbb{R}$, nous obtiendrons toujours :

$$\left.\begin{array}{l} \text{grad } P(X^*) = \rho, \quad \rho \in \mathbb{F}^n, \quad \rho \neq 0, \\ \|\text{grad } F(X^*)\| = r, \quad r \in \mathbb{F}, \quad r \neq 0, \end{array}\right\} \tag{6}$$

car les calculs sont effectués en virgule flottante et que la propagation des erreurs de calcul affecte tout résultat fourni par la machine.

Or, si les valeurs de $\rho$ ou $r$ ne représentent que l'effet cumulé des erreurs de calcul, ils sont non significatifs, et doivent être considérés comme des zéros mathématiques. Ainsi, le test d'arrêt optimal, pour tout algorithme d'optimisation de fonctions non contraintes, peut être défini de la façon suivante :

– a chaque itération $q$, on calcule le nombre de chiffres décimaux significatifs exacts de chaque composante $\rho_i^{(q)}$ de $\rho$, $i = 1, \ldots, n$ ou celui de $r^{(q)}$, soit $C_i^{(q)}$ ou $K^{(q)}$ ces nombres;

– si pour tout $i = 1, \ldots, n$ :

$$C_i^{(q)} < 1 \qquad \text{ou} \qquad K^{(q)} < 1, \tag{7}$$

alors on a atteint une solution informatiquement satisfaisante et le processus itératif doit être arrêté;

– si l'un quelconque des :

$$C_i^{(q)} \geqq 1 \qquad \text{ou} \qquad K^{(q)} \geqq 1, \tag{8}$$

alors le processus itératif doit être poursuivi.

Il faut donc, pour pouvoir utiliser ce test d'arrêt optimal, avoir un outil permettant de calculer pour toute valeur de $X \in \mathbb{F}^n$ le nombre de chiffres significatifs exacts de toutes fonctionnelles dépendant de $X$. Cet outil est le logiciel de permutation-permutation décrit dans [10] et [3] dont nous exposons au chapitre suivant les idées théoriques de base.

### 2.2. Optimisation contrainte

Lorsque l'on implémente sur ordinateur un algorithme quelconque permettant de résoudre $\inf_{\sup} \mathscr{F}(\mathscr{X})$ sous les contraintes définies par (2), c'est en fait

le problème $\inf_{\sup} F(X)$ sous les contraintes :

$$\left.\begin{array}{ll} H_i(X)=0, & i=1, \ldots, m, \quad m\in \mathbb{N}, \\ G_j(X)\geqq 0, & j=m+1, \ldots, p, \quad p\in \mathbb{N}, \end{array}\right\} \tag{9}$$

$H_i$ et $G_j$ étant les images informatiques de $h_i$ et $g_j$.

Pour cette classe de problèmes, si $X^* \in \mathbb{F}^n$ est l'image informatique de $\mathscr{X}^* \in \mathbb{R}^n$ la solution mathématique, et si les fonctions $F$, $H$ et $G$, d'une part satisfont aux conditions du 1er ordre, c'est-à-dire sont une fois différentiables et, d'autre part, satisfont à la qualification du 1er ordre des contraintes, c'est-à-dire sont telles que :

$$\left.\begin{array}{ll} \text{grad } H_i(X^*), & i=1, \ldots, m. \\ \text{grad } G_j(X^*), & j=m+1, \ldots, p, \end{array}\right\} \tag{10}$$

soient linéairement indépendants, alors on sait qu'il existe des multiplicateurs de Lagrange $\lambda$ et $\mu$ tels que les conditions de Kuhn et Tucker sont satisfaites. Ces conditions sont définies par :

$$\left.\begin{array}{ll} H_i(X^*)=0, & i=1, \ldots, m, \\ G_j(X^*)>0, & j=m+1, \ldots, p, \\ \lambda_j . G_j(X^*)=0, & j=m+1, \ldots, \text{p}, \\ \mu_i \geqq 0, & i=1, \ldots, m, \\ \nabla L(X^*, \lambda, \mu)=0, & \end{array}\right\} \tag{11}$$

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

Ainsi, à la solution $X^*$, doit-on avoir :

$$\text{grad } L(X^*, \lambda, \mu)=0. \tag{12}$$

Il est toutefois bien préférable de n'avoir dans (11) à vérifier que des contraintes d'égalité. C'est la raison pour laquelle J. Abadie [1] a proposé de transformer toutes les contraintes d'inégalités en contraintes d'égalités, en ajoutant dans ces contraintes des variables supplémentaires $X_{m+1, i}$.

Ainsi, chacune des contraintes d'inégalités du type :

$$G_j(X)>0, \qquad j=m+1, \ldots, p,$$

s'écrit-elle :

$$G_j(X) - X_{m+1,j} = 0, \tag{13}$$

avec $X_{m+1,j} \geqq 0$.

Ainsi, à la solution $X^*$ les conditions (14) sont satisfaites :

$$\left.\begin{aligned} H_i(X^*) &= 0, \qquad i = 1, \ldots, m, \\ G_j(X^*) - X^*_{m+1,j} &= 0, \qquad j = m+1, \ldots, p, \\ \lambda_j G_j(X^*) &= 0, \qquad j = m+1, \ldots, p, \\ \mu_i &\geqq 0, \qquad i = 1, \ldots, m, \\ \text{grad } L(X^*, \lambda, \mu) &= 0. \end{aligned}\right\} \tag{14}$$

Du point de vue mathématique, le processus itératif doit être interrompu lorsque les relations (14) sont satisfaites. Mais en informatique même si l'on fait calculer sur ordinateur les relations (14) avec $X^* \in \mathbb{F}^n$ image informatique de la solution mathématique, on ne trouvera jamais zéro pour les relations d'égalité (14) à cause de la propagation des erreurs d'arrondi de l'arithmétique virgule flottante. C'est la raison pour laquelle les tests d'arrêts classiques consistent à arrêter le processus itératif sur une valeur $X_s \in \mathbb{F}^n$ lorsque les valeurs des relations d'égalités sont inférieures aux $\varepsilon$ arbitrairement choisis. Mais ces tests d'arrêt sont mauvais car, comme nous l'avons dit précédemment, on ne sait pas choisir les valeurs de $\varepsilon$.

Le seul test d'arrêt satisfaisant est le test d'arrêt optimal qui consiste ici à interrompre le processus itératif sur la valeur $X_s$ pour laquelle les nombres de chiffres décimaux significatifs des relations d'égalités (14) sont tous inférieurs à 1 et ceux des relations d'inégalités (14) sont tous supérieurs à 1.

Ainsi, lorsque le code utilisé fournit à chaque itération $q$, d'une part, les valeurs du vecteur $X^{(q)}$ et, d'autre, celles des vecteurs $\lambda^{(q)}$ et $\mu^{(q)}$ alors, pour arrêter le processus itératif, il suffit de calculer les $CL_i$ de toutes les relations (14).

Si, pour toutes les relations d'égalités, tous les $CL_i < 1$ et pour toutes les relations d'inégalité, $CL_i > 1$, alors $X_s = X^{(q)}$ est une solution informatiquement satisfaisante et le processus itératif doit être interrompu.

Des versions modifiées du test d'arrêt optimal et du contrôle du nombre de chiffres décimaux significatifs de la solution optimale, valables pour la programmation non linéaire avec ou sans contraintes qui permettent de réduire sensiblement le coût de ces procédures, ont été proposées dans (6) et (7).

Les $CL_i$ vont être calculés avec la méthode exposée ci-après.

## 3. LA MÉTHODE DE PERMUTATION-PERTURBATION

### 3.1. L'origine des erreurs d'arrondi

Tout élément $x \in \mathbb{R}$ est représenté en machine par un élément $X \in \mathbb{F}$, où $\mathbb{F}$ est l'ensemble des flottants normalisés représentables en machine. Cette valeur X n'est qu'une approximation de $x$.

En fait, $x$ est écrit en virgule flottante normalisée sous la forme $x = m \times b^e$ où $m$ est une mantisse de longueur illimitée et telle que $1/b \leqq m < 1$, $e$ un exposant entier relatif, et $b$ la base.

La représentation machine $X$ de $x$ est mise sous la forme $X = M^* b^E$ où $M$ est une mantisse limitée à $p$ digits et $E$ l'exposant.

Les erreurs numériques sont essentiellement dues à cette représentation limitée des nombres en machine et elles apparaissent après chaque opération élémentaire.

Une conséquence immédiate est que l'arithmétique virgule flottante ne respecte pas les règles de l'algèbre, telle par exemple l'associativité de l'addition algébrique.

### 3.2. La méthode de Permutation

Considérons un algorithme fini quelconque, c'est-à-dire un algorithme qui en un nombre fini de calculs fournit du point de vue mathématique des résultats exacts. Cet algorithme peut être ici le calcul de $\mathscr{F}(\mathscr{X})$, $\| \operatorname{grad} \mathscr{F}(\mathscr{X}) \|$, $h_i(\mathscr{X})$, $g_j(\mathscr{X})$ ou de toute autre fonctionnelle. Cet algorithme peut être défini par :

$$\text{Procédure } (d, r, +, -, \times, :, \text{fonct}), \tag{15}$$

$d$ étant l'ensemble des données, $d \subset \mathbb{R}$, $r$ les résultats cherchés; $r \subset \mathbb{R}$.

Pour simplifier l'exposé, nous supposons $r$ unique, $r \in \mathbb{R}$. $+$, $-$, $\times$, :, fonct sont les opérateurs arithmétiques exacts.

Mise en œuvre sur ordinateur, c'est-à-dire écrite sous une forme syntaxique équivalente à la forme algébrique, cette procédure s'écrit :

$$\text{PROCEDURE } (D, R, \oplus, \ominus, *, /, \text{FONCT}), \tag{16}$$

avec $D \subset \mathbb{F}$ et $R \in \mathbb{F}$.

$\oplus$, $\ominus$, *, /, FONCT sont les opérateurs en arithmétique virgule flottante.

Puisque les règles de l'algèbre ne sont plus vérifiées, toutes les procédures informatiques correspondant à toutes les permutations possibles des opérateurs permutables seront chacune aussi représentative de la procédure algèbrique.

Aussi, si l'on appelle $C_{op}$ le nombre total des combinaisons correspondant aux diverses permutations possibles des opérateurs, on peut dire qu'il existe un ensemble $\mathscr{P}$ des procédures informatiques $P_i$ images de la seule procédure algébrique tel que :

$$\text{Card } \mathscr{P} = C_{op}. \tag{17}$$

### 3.3. La méthode de perturbation

Considérons maintenant l'une de ces images $P_i$ que l'on fait exécuter par l'ordinateur. Comme au niveau de chaque opération arithmétique les opérateurs virgule flottante engendrent une erreur d'arrondi on doit admettre que chaque opération arithmétique a deux résultats, l'un par défaut et l'autre par excès, chacun d'eux représentant aussi légitimement le résultat arithmétique exact. Par conséquent, si l'algorithme nécessite l'exécution de $k$ opérations arithmétiques (assignation, opérations arithmétiques, fonctions), il existe un ensemble $\tau$ d'éléments $\tau_i \in \mathbb{F}$ résultats informatiques représentant tous aussi légitimement le résultat exact $r$, et nous avons :

$$\text{Card } \tau = 2^k. \tag{18}$$

### 3.4. — La méthode de permutation-perturbation

Pour obtenir l'ensemble $\mathscr{R}$ des résultats informatiques images du résultat unique $r$ d'un algorithme algébrique donné, il suffit d'appliquer la méthode de perturbation à chacun des algorithmes informatiques fournis par la méthode de permutation.

Le nombre d'éléments de la population $\mathscr{R}$ ainsi obtenue est égal à :

$$\text{Card } \mathscr{R} = 2^k C_{op}. \tag{19}$$

Il existe donc Card $\mathscr{R}$ images informatiques du résultat unique $r$ d'une procédure algébrique.

### 3.5. Évaluation statistique de l'erreur sur le résultat

Si $R_i$ est un élément quelconque de $\mathscr{R}$, le nombre $C_i$ de chiffres décimaux significatifs du résultat peut être calculé par :

$$C_i = \text{Log}_{10} \frac{|R_i|}{\varepsilon_i}, \tag{20}$$

avec :

$$\varepsilon_i = \sqrt{(R_i - \overline{R})^2 + \delta^2},$$

où $\varepsilon_i$ est l'évaluation statique de l'erreur commise sur $R_i$, $\overline{R}$ étant la moyenne des éléments de la population $\mathscr{R}$, $\delta^2$ étant la variance de cette population.

Maillé [5] a montré que le meilleur estimateur de $R_i$ est donné par sa valeur moyene $\overline{R}$. Le nombre $C$ de chiffres décimaux significatifs de $\overline{R}$ est donné par :

$$C = \mathrm{Log}_{10} \frac{|\overline{R}|}{\delta}. \tag{21}$$

Toujours dans [5], il est montré que les résultats $R_i$ peuvent être considérés comme une variable aléatoire Gaussienne de moyenne $\overline{R}$ et d'écart-type $\delta$.

Nous écrirons :

$$R_i = \overline{R} + (R_i - \overline{R}) = S + B_i, \tag{22}$$

avec $B_i$, variable aléatoire Gaussienne centrée d'écart-type $\delta$; $B_i$ est un bruit numérique lié au signal $S$ du résultat théorique exact $r$.

L'intervalle de confiance à $p$ % du résultat $r$ exact défini par la loi de Student est donné par :

$$P\left[ r \in \left[ \overline{R} - t_p \frac{\delta}{\sqrt{N}}, \overline{R} + t_p \frac{\delta}{\sqrt{N}} \right] \right] = 1 - \frac{P}{100}, \tag{23}$$

où $t_p$ est la valeur de la distribution de Student pour $(N-1)$ degrés de liberté.

Il a été montré dans [5] que dans la pratique il suffit de considérer une sous-population de 3 éléments $R_i$ pour estimer $\overline{R}$.

Pour $N=3$ et $p=5$, $t_p=4{,}3$ (table de Student). L'intervalle de confiance est alors défini par $r \in I_c$, où :

$$I_c = [\overline{R} - 2{,}48\,\delta,\ \overline{R} + 2{,}48\,\delta]. \tag{24}$$

Le nombre de chiffres décimaux significatifs exact $C_e$ de $\overline{R}$ est égal à :

$$C_e = \mathrm{Log}_{10} \frac{|\overline{R}|}{e} \qquad \text{avec} \quad \frac{e}{2} = 2{,}48 \times \delta, \tag{25}$$

soit :

$$C_e = \mathrm{Log}_{10} \frac{|\overline{R}|}{\delta} - 0{,}69\ldots = C - 0{,}69. \tag{26}$$

Ainsi obtient-on avec une probabilité de 95 % le nombre de chiffres décimaux significatifs exacts de $\overline{R}$.

Dans la deuxième partie de cet article, nous présentons le logiciel de permutation-perturbation qui nous permet de calculer les trois éléments $R_i$ ainsi que le nombre de chiffres décimaux significatifs de n'importe quel résultat fourni par la procédure informatique.

## IV. LES LOGICIELS D'OPTIMISATION DE FONCTIONS NON CONTRAINTES

Tous logiciel d'optimisation d'une fonction non contrainte fait appel à un ou plusieurs sous-programmes permettant de calculer à chaque itération $q$ un certain nombre de fonctionnelles telles $F(X^{(q)})$, $\| \text{grad } F(X^{(q)}) \|$, etc.

En outre, il permet de former une suite $F(X^{(q)})$, soit monotonement décroissante dans le cas d'optimisation d'une fonction convexe, soit monotonement croissante dans le cas d'optimisation d'une fonction concave.

Dans la deuxième partie de cet article, nous présentons le logiciel écrit en FORTRAN de l'algorithme d'optimisation créé par l'auteur et décrit en détail dans [9], dans lequel sont implémentés le test d'arrêt optimal et l'évaluation de la précision de la solution fournie par la machine. Afin de faciliter la lecture de ce logiciel, nous rappelons brièvement les idées de base de la méthode d'optimisation proposée qui permet de résoudre le problème Inf $\mathscr{F}(\mathscr{X})$, $\mathscr{F}(\mathscr{X})$ se présentant sous la forme d'une somme de carrés.

### 4.1. Méthode d'optimisation de Vignes

#### 4.1.1. *Introduction*

Le but de cette méthode est d'allier la sûreté de la méthode de la plus grande pente, dans la décroissance de $\mathscr{F}(\mathscr{X})$, à la convergence quadratique de la méthode de Newton. De plus, le problème du blocage éventuel de l'algorithme sur une arête de l'hypersurface représentant la fonction de plusieurs variables à minimiser est considéré.

Pour réaliser ces objectifs, à chaque itération, un choix sera fait entre plusieurs directions.

La méthode a été conçue pour traiter le problème de la régression non linéaire et, d'une façon large, le problème d'énoncé suivant :

Étant donné $N$ valeurs $y_i$, $i=1, \ldots, N$ et $N$ fonctions de $n$ variables $g_i(x_1, \ldots, x_n)$, $i=1, \ldots, N$ avec $N \geqq n$, soit à déterminer les valeurs $x_1, \ldots, x_n$ qui rendent

minimale la fonction :

$$\mathscr{F}(x_1, \ldots, x_n) = \sum_{i=1}^{N} \{ y_i - g_i(x_1, \ldots, x_n) \}^2. \tag{27}$$

Nous considérons ici le cas où $N = n$ et $y_i = 0, i = 1, \ldots, N$. Nous avons donc à calculer : inf $\mathscr{F}(x_1, \ldots, x_n)$ avec :

$$\mathscr{F}(x_1, \ldots, x_n) = \sum_{i=1}^{n} f_i^2(x_1, \ldots, x_n). \tag{28}$$

4.1.2. *Principe*

(*a*) Soit $\mathscr{X} = (x_1, \ldots, x_n)$ un point donné de $\mathbb{R}^n$. Considérons $\mathscr{X}$ comme une approximation de la solution $\mathscr{X}^* = (x_1^*, \ldots, x_n^*)$ du problème et posons :

$$x_i^* = x_i + \Delta x_i, \qquad i = 1, \ldots, n.$$

Les développements en série de Taylor des fonctions $f_k(x_1, \ldots, x_n)$ limités au 1er ordre, s'écrivent :

$$f_k(\mathscr{X}^*) = f_k(\mathscr{X}) + \sum_{j=1}^{n} \Delta x_j \frac{\partial f_k}{\partial x_j}(\mathscr{X}) + \ldots, \qquad k, j = 1, \ldots, n$$

et (29)

$$\mathscr{F}(\mathscr{X}^*) = \sum_{k=1}^{n} \left\{ f_k(\mathscr{X}) + \sum_{j=1}^{n} \Delta x_j \frac{\partial f_k}{\partial x_j}(\mathscr{X}) + \ldots \right\}^2$$

$\mathscr{X}$ étant fixé, la quantité précédente dépend des $\Delta x_i$ et à l'extremum nous aurons :

$$\frac{\partial \mathscr{F}}{\partial \Delta x_i} = 0, \qquad i = 1, \ldots, n, \tag{30}$$

soit, en se limitant au développement du 1er ordre :

$$\sum_{k=1}^{n} \left\{ f_k(\mathscr{X}) + \sum_{j=1}^{n} \Delta x_j \frac{\partial f_k}{\partial x_j}(\mathscr{X}) \right\} \frac{\partial f_k}{\partial x_i}(\mathscr{X}) = 0. \tag{31}$$

Le vecteur correction $\Delta\mathscr{X} = (\Delta x_1, \ldots, \Delta x_n)$ est obtenu en résolvant le système linéaire (31). Ce système linéaire se met sous la forme :

$$Z . \Delta\mathscr{X} = C, \tag{32}$$

avec :

$$Z_{ij} = \sum_{k=1}^{n} \frac{\partial f_k}{\partial x_i}(\mathscr{X}) \frac{\partial f_k}{\partial x_j}(\mathscr{X}),$$

$$C_i = - \sum_{k=1}^{n} f_k(\mathscr{X}) \frac{\partial f_k}{\partial x_i}(\mathscr{X}),$$

ou, sous forme développée :

$$Z = \begin{bmatrix} \sum_k \frac{\partial f_k}{\partial x_1}\frac{\partial f_k}{\partial x_1} \cdots \sum_k \frac{\partial f_k}{\partial x_1}\frac{\partial f_k}{\partial x_n} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \sum_k \frac{\partial f_k}{\partial x_n}\frac{\partial f_k}{\partial x_1} \cdots \sum_k \frac{\partial f_k}{\partial x_n}\frac{\partial f_k}{\partial x_n} \end{bmatrix},$$

$$C = -\begin{bmatrix} \sum_k f_k \frac{\partial f_k}{\partial x_1} \\ \vdots \\ \sum_k f_k \frac{\partial f_k}{\partial x_n} \end{bmatrix}, \tag{33}$$

$Z$ est une matrice symétrique.

De plus, si $\mathcal{J}$ désigne la matrice jacobienne des fonctions $f_i$, $i = 1, 2, \ldots, n$, évaluée au point $\mathcal{X}$, soit :

$$\mathcal{J} = \left[\frac{\partial f_i}{\partial x_j}(\mathcal{X})\right] \tag{34}$$

et si $\varphi$ est le vecteur de $\mathbb{R}^n$ de $i$-ième composante $f_i(\mathcal{X})$, soit :

$$\varphi = [f_i(\mathcal{X})],$$

on a :

$$Z = \mathcal{J}^t \mathcal{J} \qquad \text{et} \qquad C = -\mathcal{J}^t \varphi, \tag{35}$$

(32) s'écrit donc :

$$\mathcal{J}^t \mathcal{J} \,.\, \Delta \mathcal{X} = -\mathcal{J}^t \varphi \tag{36}$$

et si $\mathcal{D}et\, J \neq 0$, ce système est équivalent au système obtenu par la méthode de Newton :

$$\mathcal{J}\, \Delta\mathcal{X} = -\varphi. \tag{37}$$

Toutefois, la correction d'un vecteur $\mathcal{X}$ par l'addition de $\Delta\mathcal{X}$ solution de (32) ou (37) n'assure pas toujours que la relation (38) est satisfaite :

$$\mathcal{F}(\mathcal{X} + \Delta\mathcal{X}) < \mathcal{F}(\mathcal{X}). \tag{38}$$

Ainsi, diminuer la valeur de $\mathcal{F}(\mathcal{X})$ à chaque itération est le but de l'algorithme considéré que nous exposons en (*c*); auparavant, dans le paragraphe qui suit, nous envisagerons brièvement les difficultés dues à l'existence d'arêtes.

(*b*) *La présence d'une arête*

Soit (*S*) l'hypersurface d'équation : $z = \mathscr{F}(x_1, \ldots, x_n)$.

Nous définirons la notion d'arête de (*S*) au sens mathématique comme étant le lieu des points où l'une des composantes au moins de grad $\mathscr{F}$ est discontinue.

Dans la pratique, il est assez rare de rencontrer des surfaces possédant des arêtes au sens strict du terme, mais il est courant de rencontrer des pseudo-arêtes, lieu des points où l'une au moins des composantes de grad $\mathscr{F}$ subit des variations très rapides.

Nous utiliserons alors le mot « arête » pour désigner indifféremment une arête ou une pseudo-arête.

La présence d'une arête bloque souvent le procédé calculant le minimum de $\mathscr{F}(x_1, \ldots, x_n)$. Il convient dans ce cas :

- de découvrir la présence de l'arête;
- de suivre son coutour pour atteindre l'extremum cherché.

La détection se fera selon le procédé suivant :

Deux itérés successifs $\mathscr{X}^{(q-1)}$ et $\mathscr{X}^{(q)}$ étant calculés, on examine la nature de l'angle $\theta$ que font entre eux les vecteurs grad $\mathscr{F}_{q-1}$ et grad $\mathscr{F}_q$.

($\alpha$) Si $\theta$ est aigu, c'est-à-dire si la condition :

$$\cos\theta > 0 \qquad \text{ou} \qquad (\text{grad}\,\mathscr{F}_{q-1}, \text{grad}\,\mathscr{F}_q) > 0, \tag{39}$$

est vérifiée, on conclura que la fonction ne présente pas d'arête, ou bien en présente une, mais que les points $\mathscr{X}^{(q-1)}$ et $\mathscr{X}^{(q)}$ sont du même côté; il n'y aura pas lieu alors de considérer pour la suite une direction particularisée de déplacement.

($\beta$) Si $\theta$ est obtus ou droit, c'est-à-dire si la condition :

$$\cos\theta \leqq 0 \qquad \text{ou} \qquad (\text{grad}\,\mathscr{F}_{q-1}, \text{grad}\,\mathscr{F}_q) \leqq 0, \tag{40}$$

est vérifiée, on conclura que la fonction présente une arête et que les points $\mathscr{X}^{(q-1)}$ et $\mathscr{X}^{(q)}$ sont de part et d'autre; dans ce cas, on cherchera à se déplacer dans une direction sensiblement parallèle à l'arête qui sera la direction du vecteur $w$ de définition :

$$w = u^{(q-1)} + u^{(q)}$$

$$\text{où} \quad u^{(q-1)} = -\frac{\text{grad}\,\mathscr{F}_{q-1}}{\|\text{grad}\,\mathscr{F}_{q-1}\|}, \qquad u^{(q)} = -\frac{\text{grad}\,\mathscr{F}_q}{\|\text{grad}\,\mathscr{F}_q\|}. \tag{41}$$

(*c*) *Passage d'un itéré à l'itéré d'ordre supérieur*

L'algorithme consiste, à partir d'un élément initial quelconque $\mathscr{X}^{(0)}$ de $\mathbb{R}^n$, à former une suite d'éléments $\{\mathscr{X}^{(q)}\}$ de $\mathbb{R}^n$, le passage de $\mathscr{X}^{(q)}$ à $\mathscr{X}^{(q+1)}$ s'effectuant

par un choix entre les directions suivantes :

– la direction $\Delta\mathscr{X}^{(q)}$ donnée par la résolution du système (37);

– la direction de $-\text{grad}\,\mathscr{F}_q$;

– s'il y a lieu, la direction $w^{(q)}$ définie après détection d'une arête (41),

de façon à réaliser :

1. $$\mathscr{F}(\mathscr{X}^{(q+1)}) < \mathscr{F}(\mathscr{X}^{(q)}), \tag{42}$$

2. $$\mathscr{F}(\mathscr{X}^{(q)}) - \mathscr{F}(\mathscr{X}^{q+1}) \text{ maximum.} \tag{43}$$

Pour cela :

(α) Les directions considérées sont déterminées à chaque pas.

(β) On cherche ensuite, dans chacune des directions, le point de $\mathbb{R}^n$ où $\mathscr{F}(\mathscr{X})$ prend la valeur minimale :

– Si la direction est celle de $-\text{grad}\,\mathscr{F}_q$ ou de $w^{(q)}$, cette recherche est effectuée grossièrement selon les processus classiques. On prend soin de normer le vecteur définissant la direction de recherche et, après l'obtention de ce vecteur unitaire $\tau$, on calcule des valeurs $\mathscr{F}(\mathscr{X}^{(q)}+\alpha\tau)$ avec $\alpha=1$ puis, selon les cas, des valeurs de $\alpha$ multipliées par 2 ou divisées par 2.

– Si la direction est celle de $\Delta\mathscr{X}^{(q)}$, on ne norme pas le vecteur définissant la direction et on pose $\tau=\Delta\mathscr{X}^{(q)}$, $\alpha=1$ puis, si $\mathscr{F}(\mathscr{X}^{(q)}+\alpha\tau)<\mathscr{F}(\mathscr{X}^{(q)})$, on considérera comme satisfaisant le point $\mathscr{X}^{(q)}+\Delta\mathscr{X}^{(q)}$.

– Si $\mathscr{F}(\mathscr{X}^{(q)}+\alpha\tau)\geqq\mathscr{F}(\mathscr{X}^{(q)})$, on divisera par 2 la valeur de $\alpha$ jusqu'à obtention d'un point $\mathscr{X}^{(q)}+\alpha\tau$ satisfaisant l'égalité $\mathscr{F}(\mathscr{X}^{(q)}+\alpha\tau)<\mathscr{F}(\mathscr{X}^{(q)})$.

(γ) Trois points au maximum ont été ainsi retenus sur l'ensemble des directions considérées. On choisit alors parmi les points, le point $\mathscr{Y}$ qui réalise :

$$\mathscr{F}(\mathscr{X}^{(q)}) - \mathscr{F}(\mathscr{Y}) \text{ maximum} \tag{44}$$

et l'on pose :

$$\mathscr{X}^{(q+1)} = \mathscr{Y}. \tag{45}$$

On pourra se reporter à [9] pour la démonstration de convergence de l'algorithme.

## 5. L'IMPLÉMENTATION DU TEST D'ARRÊT OPTIMAL

### 5.1. Pour les méthodes d'optimisation non contrainte

L'implémentation du test d'arrêt optimal consiste, à chaque itération $q$, à calculer les nombres $C_i^{(q)}$ ou $K^{(q)}$ de chiffres décimaux significatifs de chaque composante de grad $F(X^{(q)})$ ou de $\|\text{grad}\,F(X^{(q)})\|$.

Le calcul de $C_i^{(q)}$, $i=1, 2, \ldots, n$ ou $K^{(q)}$ se fait en calculant trois valeurs de chaque composante de grad $F(X^{(q)})$ ou 3 valeurs de $\| \text{ grad } F(X^{(q)}) \|$ à l'aide du logiciel PEPER puis, en calculant leur moyenne, leur écart-type et enfin $C_i^{(q)}$ ou $K^{(q)}$ à l'aide de l'équation (26).

Le ou les sous-programmes de calcul des composantes de grad $F(X^{(q)})$ ou de $\| \text{ grad } F(X^{(q)}) \|$ doivent être structurés, comme il est montré dans la deuxième partie de cet article, afin de pouvoir utiliser le logiciel PEPER de permutation-perturbation.

Lorsque tous les $C_i^{(q)} < 1, i=1, 2, \ldots, n$, ou lorsque $K^{(q)} < 1$, alors le processus itératif est interrompu, sinon les itérations sont poursuivies.

Le test d'arrêt optimal permet donc d'interrompre le processus itératif dès qu'une solution informatiquement satisfaisante est atteinte, c'est-à-dire que l'on fait juste le nombre d'itérations nécessaires.

## 5.2. Pour les méthodes d'optimisation contrainte

Nous avons vu que le test d'arrêt optimal consiste à calculer le nombre de chiffres décimaux significatifs $CL_i$ de chaque relation (14) et à interrompre le processus itératif lorsque tous les $CL_i < 1$ pour les relations d'égalités et les $CL_i > 1$ pour les relations d'inégalités. Parmi ces relations (14) figurent les contraintes qui peuvent être soit linéaires soit non linéaires.

5.2.1. *Les contraintes linéaires*

(*a*) *Les contraintes d'égalité*

Ces contraintes se présentent mathématiquement sous la forme :

$$h_i = \sum_{l=1}^{n} a_{il} . x_l - k_i = 0,$$

$$a_{il}, x_l, k_i \in \mathbb{R}, \qquad i=1, \ldots, t, \qquad t \in \mathbb{N}. \tag{46}$$

Considérons le vecteur $\mathscr{X}_s \in \mathbb{R}^n$ qui satisfait exactement (46) et $X_s \in \mathbb{F}^n$, la représentation machine de $\mathscr{X}_s$. Le calcul sur ordinateur de (46) donnera toujours :

$$H_i = \sum_{l=1}^{n} A_{il} * X_{s_l} - K_i = \rho_i, \qquad \rho_i \neq 0. \tag{47}$$

En effet, le résultat du calcul de ce produit scalaire est toujours entaché par la propagation des erreurs de calcul et, de ce fait, la stricte égalité à zéro n'est

pratiquement jamais satisfaite. Or, si $\rho_i, i=1, \ldots, t$ ne représente que l'effet des erreurs cumulées de calcul, alors on peut affirmer que $\rho_i$ est un zéro et que par conséquent les contraintes d'égalité sont vérifiées.

Or il a été donné dans [3] et [4] une estimation de l'erreur de calcul dans un produit scalaire. Ce résidu théorique est donné par la relation :

$$\hat{\rho}_i = 2^{-pb} \sqrt{n^u \sum_{l=1}^{n} (A_{i_l} * X_{s_l})^2 + K_i^2}, \tag{48}$$

où $pb$ est le nombre de bits de la mantisse, $u=1$ si l'arithmétique est en arrondi, $u=2$ si l'arithmétique est en troncature.

En toute rigueur, $n$ est le nombre de monomes non nuls dans $H_i$.

En utilisant (49), il est facile de vérifier si les contraintes linéaires d'égalités sont satisfaites. En effet, si :

$$\rho_i^* = \frac{|\rho_i|}{\hat{\rho}_i} \sim 1, \qquad i=1, \ldots, t, \tag{49}$$

alors toutes les contraintes $H_i$ sont vérifiées. Si l'un ou plusieurs $\rho_i^*$ ne satisfont pas (49), alors les contraintes ne sont pas satisfaites.

(*b*) *Les contraintes d'inégalité*

Ces contraintes se présentent mathématiquement sous la forme :

$$h_i = \sum_{l=1}^{\cdot\cdot} a_{i_l} x_l - k_i > 0, \qquad i=1, \ldots, t. \tag{50}$$

La vérification de ces contraintes linéaires d'inégalité se fait de la façon suivante :

$$\text{Si} \quad \rho_i^* = \frac{|\rho_i|}{\hat{\rho}_i} > 1, \qquad i=1, \ldots, t, \tag{51}$$

alors les contraintes d'inégalité sont vérifiées. Si l'inéquation (51) n'est pas satisfaite, elles ne sont pas vérifiées.

### 5.2. Les contraintes non linéaires

(*a*) *Les contraintes d'égalité*

Ces contraintes se présentent sous la forme définie par :

$$h_i(\mathscr{X}) = 0, \qquad i=1, \ldots, t, \tag{52}$$

où les $h_i$ ont des structures algébriques absolument quelconques.

Comme il a été dit précédemment, la seule méthode satisfaisante pour vérifier ces contraintes d'égalité est de calculer par le logiciel de permutation-perturbation le nombre $CL_i$ de chiffres décimaux significatifs de $H_i(X_s)$.

$$\text{Si} \quad CL_i < 1, \qquad i = 1, \ldots, t, \tag{53}$$

alors les contraintes sont vérifiées.

$$\text{Si} \quad CL_i \geqq 1, \tag{54}$$

pour certains $i \in [1, t]$, alors elles ne sont pas satisfaites.

(*b*) *Les contraintes d'inégalité*

Ces contraintes se présentent sous la forme définie par :

$$h_i(x) > 0, \qquad i = 1, \ldots, t. \tag{55}$$

Comme il est dit précédemment, la seule méthode satisfaisante pour vérifier ces contraintes est que les inégalités définies par (54) soient satisfaites pour tout $i$.

Si certains $CL_i < 1$, alors les contraintes d'inégalité ne sont pas vérifiées.

## 6. ÉVALUATION DE LA PRÉCISION

Nous avons vu jusqu'à maintenant que l'utilisation de la méthode de permutation-perturbation permettait d'arrêter le processus itératif dans les méthodes d'optimisation contraintes ou non contraintes dès qu'une solution informatique est atteinte.

Mais il faut remarquer, comme il a été montré dans [3], qu'une solution peut être informatiquement satisfaisante sans pour cela être une bonne approximation de la solution mathématique. C'est la raison pour laquelle il est absolument indispensable d'évaluer la précision de la solution informatique. Pour ce faire il suffit, comme il a été montré dans [3] et [10], de faire exécuter trois fois le programme en mettant en œuvre les tests d'arrêts optimaux qui utilisent le logiciel de permutation-perturbation. Ainsi, on obtient 3 vecteurs solutions $X_{sj}$, $j = 1, 2, 3$; et en utilisant toujours l'équation (26), nous en déduisons le nombre de chiffres décimaux significatifs de chacune des composantes de la solution $\overline{X}_s$ moyenne des $X_{sj}, j = 1, 2, 3$. Des exemples sont donnés dans la deuxième partie de cet article.

## 7. CONCLUSION

Nous avons présenté dans la première partie de cet article une méthodologie permettant de résoudre les problèmes qui se posent lorsque l'on implémente sur ordinateur des méthodes d'optimisation.

Tout d'abord, l'utilisation des tests d'arrêts optimaux permettent, pour les méthodes d'optimisation contrainte ou non contrainte, d'interrompre le processus itératif dès qu'une solution informatique satisfaisante est atteinte.

De plus, l'utilisation de la méthode de permutation-perturbation appliquée aux algorithmes d'optimisation eux-mêmes permet d'estimer la précision de la solution fournie par l'ordinateur.

## BIBLIOGRAPHIE


1. J. ABADIE, *The GRG Method for Nonlinear Programming Design and Implementation of Optimization Software*, in HARVEY J. GREENBERG, éd., Sijthoff et Noordhoff, 1978, p. 335-362.
2. D. M. HIMMELBLAU, *Numerical Methods for Nonlinear Optimization*, in F. A. LOOTSMA, ed., Academic Press, 1972, p. 69-73.
3. M. LA PORTE et J. VIGNES, *Algorithmes numériques, analyse et mise en œuvre*, t. 1; Technip, éd., 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.