RAIRO. RECHERCHE OPÉRATIONNELLE

J. ABADIE
Y. SALHI

Sur la dérivation numérique en optimisation

RAIRO. Recherche opérationnelle, tome 22, n° 1 (1988), p. 1-26
<http://www.numdam.org/item?id=RO_1988__22_1_1_0>

Recherche opérationnelle/Operations Research
(vol. 22, n° 1, 1988, p. 1 à 26)

SUR LA DÉRIVATION NUMÉRIQUE EN OPTIMISATION (*)

par J. ABADIE [1], Y. SALHI [1]

Résumé. — *La méthode de Newton pour la minimisation d'une fonction de n variables possède une qualité incomparable, sa convergence finale quadratique sous des hypothèses très larges, et quelques défauts, dont une convergence non assurée si le point de départ est loin de la solution souhaitée, et la nécessité de calculer les dérivées premières et secondes. Nous montrons dans cet article comment on peut s'affranchir de ces deux difficultés, en supposant cependant que les dérivées premières soient calculées par des formules exactes. Les dérivées secondes sont calculées par différences centrales des dérivées premières. Le calcul du pas h de différentiation est fait de façon à établir un bon compromis entre les deux nécessités que sont la précision et la vitesse de calcul. Des expériences numériques montrent le bien fondé de la méthode proposée.*

Mots clés : Programmation non linéaire; méthode de Newton; dérivées numériques; expériences numériques.

Abstract. — *Newton's method for function minimisation possesses an outstanding property. Its final quadratic convergence under very weak hypothesis, but also some drawbacks, among which (i) the possibility of non convergence when starting far from the solution sought and (ii) the necessity of computing (and then programming) the first and second partial derivatives. We show in this paper how to avoid both difficulties, assuming however that the first partial derivatives are computed by exact formulae. The second partial derivatives are computed by central differences of first derivatives. The difference step h is calculated so as to establish a trade-off between accuracy and computation speed. Encouraging numerical experiments support the proposed method.*

Keywords : Non linear programming; Newton's method; numerical derivatives; numerical experiments.

(*) Reçu octobre 1986.

(1) Groupe de recherche « Validité du Logiciel numérique ». Université Pierre-et-Marie-Curie, Institut de programmation, tour n° 65-66, 4, place Jussieu, Paris Cedex 05.

[Le premier auteur était Professeur à l'Université de Chicago, Graduate School of Business, lors de la rédaction de cet article].

Recherche opérationnelle/Operations Research, 0399-0559/88/01 1 26/$ 4.60

1. INTRODUCTION

Approximer une dérivée sur ordinateur, c'est mettre en œuvre sur ordinateur une formule du type

$$\lim_{h \to 0} \frac{g(x+h)-g(x-h)}{2h}.$$

On pourrait penser qu'il ne s'agit là que d'un problème de programmation. En fait pas du tout, car cette formule a été créée pour être utilisée dans le domaine du continu avec une arithmétique infiniment précise. Or lorsqu'elle est traitée sur ordinateur, elle est projetée dans le domaine du discret et exécutée avec une arithmétique à précision limitée.

Il résulte de cette transposition une dégradation de l'information au fur et à mesure de l'élaboration des calculs, dégradation qui conduit à l'obtention de résultats entachés d'erreurs et parfois même aberrants.

L'approximation des dérivées secondes se pose souvent dans les algorithmes d'optimisation utilisant la méthode de Newton. La programmation des formules analytiques de la matrice hessienne étant parfois long, fastidieux et générateur d'erreurs, surtout dans le cas de fonctions compliquées, on a souvent recours à la dérivation numérique.

Dans la suite, toutes nos fonctions seront de classe C^3 (lorsqu'elles seront de classe C^4 cela sera spécifié). Mathématiquement, dans le cas le plus général, la dérivée d'une telle fonction est donnée par:

$$\lim_{\Delta x \to 0} \frac{\psi(x+\Delta x)-\psi(x-\Delta x)}{2\Delta x} \tag{1}$$

Un développement de Taylor en $x+\Delta x$ et $x-\Delta x$ permet de montrer que la formule précédente est précise au second ordre de Δx. C'est-à-dire:

$$\psi'(x)=\frac{\psi(x+\Delta x)-\psi(x-\Delta x)}{2\Delta x}+o(\Delta x^2), \qquad \Delta x \neq 0 \tag{2}$$

où $o(\Delta x^2)=O(\Delta x^3)$ si ψ est de classe C^3.

Cela peut être effectué sur toute fonction de classe C^3, donc en particulier sur la dérivée première d'une fonction donnée afin de calculer la dérivée seconde.

Soit f une fonction de R^n dans R, supposée de classe C^4. Nous dirons: $f \in C^4$ $[R^n]$.

Notons $g(x)=\nabla f(x)=(g_1(x), g_2(x), \ldots, g_n(x))^t$ le gradient de la fonction en x et e_i le i-ème vecteur de la base canonique de R^n.

Les dérivées secondes de f peuvent être approximées en utilisant la formule (2) pour $g_i(x)$, $i=1, \ldots, n$.

Tout élément $\partial^2 f(x)/\partial x_i \partial x_j$ est approximé par:

$$d(h)=\frac{g_i(x+he_j)-g_i(x-he_j)}{2h}, \qquad h\neq 0 \tag{3}$$

où h est appelé pas de dérivation.

Pour le calcul numérique il n'est pas possible d'utiliser une formule du type (1), c'est-à-dire de passer à la limite quand h tend vers zéro. En calculant donc dans (3) avec $h\neq 0$ le résultat $d(h)$ n'est qu'une approximation de la dérivée, dont on pourra juger la validité à travers l'étude expérimentale que nous avons menée.

2. ERREUR COMMISE DANS CETTE APPROXIMATION ET PRINCIPALES CONSÉQUENCES

L'étude de ces erreurs est menée dans le cas particulier du problème d'optimisation non contrainte (elle est cependant généralisable à tout autre cas).

Nous rappelons que le problème d'optimisation non contrainte se présente en général sous la forme canonique suivante où $f\in C^k[R^n]$:

$$\underset{x\in R^n}{\text{MIN}} f(x)$$

Ce problème est en général résolu par des algorithmes itératifs. Parmi les plus connus nous citerons la méthode de plus grande pente, la méthode de Newton et les méthodes à métrique variable [2].

L'algorithme itératif utilisé dans notre étude est donné dans Salhi [10]. Cet algorithme est un compromis de trois méthodes de base:

– La méthode de Newton à pas variable.

– La méthode de plus grande pente avec recherche unidimensionnelle approchée [3].

– La méthode de Vignes pour contourner les arêtes éventuelles sur l'hypersurface représentant la fonction [14].

Le calcul de la direction de Newton z^k à l'itération k est effectué en approximant le hessien par dérivation numérique. En première expérience cette dérivation est mise en œuvre avec un pas *fixé* au début de l'algorithme, donc le même pour tout élément de la matrice hessienne et à toute itération. Elle a été réalisée sur Cray-1 et CDC Cyber 750 en simple précision avec 48 bits de mantisse soit 14 chiffres décimaux significatifs (plus précisément 14.4).

Les résultats obtenus pour quelques problèmes tests (à solution connue) sont données sous forme de tableau. Pour chacun de ces problèmes nous choisissons *trois pas* de dérivation et recherchons le minimum de la fonction f séparément pour chacun de ces pas. Les problèmes testés sont représentés par des numéros dans le tableau de résultats, les expressions analytiques correspondantes sont données en annexe.

A travers ces quelques résultats, on constate que le pas de dérivation joue un rôle important dans le bon déroulement et la convergence de l'algorithme. En choisissant des pas de dérivation différents pour diverses fonctions, nous constatons que le meilleur pas dépend intimement de la fonction testée. Il suffit de changer certaines variables ou la fonction pour changer un bon pas en un mauvais et réciproquement. Ceci peut s'expliquer si l'on procède à une analyse des erreurs commises dans l'approximation des dérivées.

En effet pour toute fonction ψ, $\psi'(x)$ est approximée par:

$$\Psi'_a(x) = \frac{\Psi(x+h) - \Psi(x-h)}{2h} \tag{4}$$

Or la dérivée exacte est donnée par:

$$\psi'(x) = \psi'_a(x) + o(h^2).$$

Dans cette approximation, il y a une première erreur $o(h^2)$ due à la troncature de la formule de Taylor, et qui est appelée erreur de méthode. Nous la noterons e_m. Cette erreur de méthode peut être rendue petite en diminuant h. Mais mise en œuvre sur ordinateur l'approximation $\psi'_a(x)$ sera, outre l'erreur de méthode, entachée d'une erreur de calcul e_c qui est d'autant plus forte que h est petit.

TABLEAU I

— *La première colonne donne le numéro du problème testé.*
— *Un pas est fixé pour chaque colonne* j, j=2, 3, 4.
— *La deuxième, troisième, quatrième colonne donnent chacune:*
• *le nombre d'itérations effectuées*: *itérations*;
• *le vecteur solution sous forme de flottants*;
• *la valeur de la fonction à l'arrêt sous forme de flottant*;
• *le nombre d'évaluations du gradient*: Ng.
— *la dernière colonne donne la solution mathématique.*

FCT	Pas de dérivation et résultats			X * et FX *
	h = 0.01	h = 0.0000001	h = 0.000000000001	
1	itérations = 42 .999999999996255E+00 .999999999988763E+00 .140229933996004E-22 Ng = 215	itérations = 32 .999999999999382E+00 .999999999998138E+00 .382136844263542E-24 Ng = 160	itérations = 71 .100000000000163E+01 .100000000000488E+01 .264759390276449E-23 Ng = 360	 1.0 1.0 .0
2	itérations = 9 .299999999999893E+01 .499999999999719E+00 .193113149598104E-24 Ng = 50	itérations = 7 .299999999999987E+01 .499999999999975E+00 .388750654092915E-26 Ng = 40	itérations = 10 .299999999999969E+01 .499999999999925E+00 .190336359211726E-25 Ng = 55	 3.0 .5 .0
3	itérations = 3 .399999999999898E+01 .899999999999903E+01 -.182000000000008E+02 Ng = 15	itérations = 3 .399999996727718E+01 .899999975411413E+01 -.182000000000008E+02 Ng = 15	itérations = 7 .400000000000000E+01 .899999999873330E+01 -.182000000000008E+02 Ng = 35	 4.0 9.0 -18.2
4	itérations = 20 .106982137304398E-13 .136813643374432E-13 .100000000000000E+01 .000000000000000E+00 Ng = 147	itérations = 18 .211441413949411E-13 .227685837816078E-13 .100000000000000E+01 .000000000000000E+00 Ng = 133	itérations = 100 ** .144177125630740E-11 .144189492722651E-11 .999999999999719E+00 .807793566946316E-24 Ng = 701	 .0 .0 1.0 .0
5	itérations = 15 .999999999999989E+00 .999999999999979E+00 .100000000000001E+01 .100000000000002E+01 .639166659846274E-26 Ng = 135	itérations = 14 .100000000000001E+01 .100000000000001E+01 .100000000000001E+01 .100000000000001E+01 .817890986533145E-26 Ng = 126	itérations = 23 .999999999999968E+00 .999999999999936E+00 .100000000000004E+01 .100000000000007E+01 .611398755982559E-26 Ng = 207	 1.0 1.0 1.0 1.0 .0
6	itérations = 100 ** .217224873684114E-04 -.217224851281978E-05 -.763654151199236E-03 -.763654151647281E-03 .921508751820560E-11 Ng = 901	itérations = 30 .173735102158172E-04 -.173735102158174E-05 -.348668671996025E-05 -.348668671996087E-05 .189428785269117E-17 Ng = 279	itérations = 100 ** .323041680888720E-03 -.323041682056026E-04 .135780911029259E-03 .135780935385383E-03 .208223253542700E-13 Ng = 901	 .0 .0 .0 .0 .0

Pour h=0.001 la fonction numéro 3 donne de meilleurs résultats:

X_1 = 0.400000000000674E+01 itérations = 2

X_2 = 0.900000000002007E+01 et

FX = - 0.181999999999999E+02 Ng = 10

** 100 est le nombre maximal d'itérations autorisées.

TABLEAU I (*suite*)

	itérations = 2	itérations = 3	itérations = 4	
	-.807793566946316E-27	.000000000000000E+00	-.986076131526265E-31	.0
	-.807793566946316E-27	.000000000000000E+00	-.986076131526265E-31	.0
	-.807793566946316E-27	.940395480657830E-37	-.197215226305253E-30	.0
	-.807793566946316E-27	.940395480657830E-37	-.197215226305253E-30	.0
	-.807793566946316E-27	.940395480657830E-37	-.197215226305253E-30	.0
	-.323117426778526E-26	.940395480657830E-37	-.986076131526265E-31	.0
	-.161558713389263E-26	.940395480657830E-37	-.986076131526265E-31	.0
11	-.484676140167790E-26	.000000000000000E+00	-.123259516440783E-31	.0
	-.403896783476315E-27	.117549435082229E-37	-.157772181044202E-29	.0
	-.242338070083895E-26	.470197740328915E-37	-.986076131526265E-31	.0
	-.323117426778526E-26	.000000000000000E+00	.000000000000000E+00	.0
	-.242338070083895E-26	.940395480657830E-37	.000000000000000E+00	.0
	-.161558713389263E-26	.000000000000000E+00	-.197215226305253E-30	.0
	-.161558713389263E-26	.000000000000000E+00	-.493038065763132E-31	.0
	-.161558713389263E-26	.000000000000000E+00	-.394430452610506E-30	.0
	-.161558713389263E-26	.000000000000000E+00	-.394430452610506E-30	.0
	-.161558713389263E-26	.000000000000000E+00	-.157772181044202E-29	.0
	.000000000000000E+00	.188079096131566E-36	.000000000000000E+00	.0
	-.807793566946316E-27	.940395480657830E-37	-.197215226305253E-30	.0
	-.807793566946316E-27	.940395480657830E-37	-.197215226305253E-30	.0
	.706136015661183E-39	.112014317342821E-58	.374988940212944E-45	.0
	Ng = 123	Ng = 164	Ng = 205	
	itérations = 3	itérations = 5	itérations = 9	
	.333333333333332E+00	.333333333333332E+00	.333333333333332E+00	1/3
	.666666666666664E+00	.666666666666664E+00	.666666666666664E+00	2/3
	.999999999999996E+00	.999999999999996E+00	.999999999999996E+00	1.0
	.133333333333333E+01	.133333333333333E+01	.133333333333333E+01	4/3
	.166666666666666E+01	.166666666666666E+01	.166666666666666E+01	5/3
	.199999999999999E+01	.199999999999999E+01	.199999999999999E+01	2.0
	.233333333333331E+01	.233333333333331E+01	.233333333333331E+01	7/3
	.266666666666666E+01	.266666666666666E+01	.266666666666666E+01	8/3
	.299999999999999E+01	.299999999999999E+01	.299999999999999E+01	3.0
12	.333333333333331E+01	.333333333333331E+01	.333333333333331E+01	10/3
	.366666666666664E+01	.366666666666664E+01	.366666666666664E+01	11/3
	.399999999999999E+01	.399999999999999E+01	.399999999999999E+01	4.0
	.433333333333331E+01	.433333333333331E+01	.433333333333331E+01	13/3
	.466666666666663E+01	.466666666666663E+01	.466666666666663E+01	14/3
	.499999999999997E+01	.499999999999997E+01	.499999999999997E+01	5.0
	.533333333333331E+01	.533333333333331E+01	.533333333333331E+01	16/3
	.566666666666663E+01	.566666666666663E+01	.566666666666663E+01	17/3
	.599999999999997E+01	.599999999999997E+01	.599999999999997E+01	6.0
	.633333333333329E+01	.633333333333329E+01	.633333333333329E+01	19/3
	.666666666666663E+01	.666666666666663E+01	.666666666666663E+01	20/3
	.000000000000000E+00	.000000000000000E+00	.000000000000000E+00	.0
	Ng = 164	Ng = 246	Ng = 410	

3. ÉTUDE DES ERREURS DE CALCUL ET DE MÉTHODE

Dans ce paragraphe nous allons étudier les erreurs de calcul et de méthode, afin de déterminer la meilleure stratégie à adopter pour les minimiser. En raison d'un souci de simplification des notations cette étude sera menée dans le cas d'une fonction numérique à une variable. Il suffit de l'appliquer alors n fois dans le cas du gradient d'une fonction $f: R^n \to R$.

Notons

$$D(H) = (\Phi(X \oplus H) \ominus \Phi(X \ominus H)) \oslash 2 \circledast H \tag{5}$$

l'approximation-machine de la dérivée d'une fonction φ où :

- Φ désigne une image de φ en machine
- X et H désignent les représentations de x et h
- $\ominus$, $\oplus$, $\circledast$ et $\oslash$ désignent les opérateurs informatiques approximant les opérateurs arithmétiques exacts correspondants.

La valeur $D(H)$ approximant $\varphi'(x)$ est entachée d'une erreur de méthode e_m et d'une erreur de calcul e_c.

Définition de l'erreur de calcul

$$\varphi'_a(x) = \frac{\varphi(x+h) - \varphi(x-h)}{2h} = \frac{\varphi(X \oplus H) - \varphi(X \ominus H)}{2H} + e_1 \tag{6}$$

e_1 représentant l'erreur absolue sur la valeur $\varphi'_a(x)$ calculée avec les représentations en machine des valeurs exactes des arguments. Mais

$$\frac{\varphi(X \oplus H) - \varphi(X \ominus H)}{2H} = \frac{\Phi(X \oplus H) - \Phi(X \ominus H)}{2H} + e_2 \tag{7}$$

où e_2 représente l'erreur engendrée par Φ. De plus

$$\frac{\Phi(X \oplus H) - \Phi(X \ominus H)}{2H} = D(H) + e_3 \tag{8}$$

e_3 étant l'erreur engendrée par les opérateurs $\ominus$, $\oslash$ et $\circledast$. Donc l'erreur de calcul e_c est égale à $e_1 + e_2 + e_3$

- L'erreur e_3 peut être négligée par rapport aux autres.
- L'erreur e_1 peut également être négligée car elle provient du remplacement de x et h par X et H, et des opérateurs $\oplus$ et $\ominus$ dans le calcul de $X \oplus H$

et $X \ominus H$. On peut choisir H de façon à rendre l'erreur dans ce calcul rigoureusement nulle. Pour cela il suffit d'avoir: $H \in I = [X^* 2^{-m}, X^* 2^{+m}]$, m étant le nombre de bits de la mantisse, de la machine utilisée. (Cela assure qu'aucun des opérateurs H et X ne disparaît devant l'autre au cours du calcul de $X+H$ et $X-H$). En calculant H par

$$H = (H_p \oplus X) \ominus X, \qquad H_p \in I,$$

on assure

$$X \oplus H = X + H \text{ et } X \ominus H = X - H.$$

Donc l'erreur de calcul e_c peut être assimilée à e_2, et l'erreur globale entachant $D(H)$ est alors

$$e_g = e_m + e_2 \tag{9}$$

Expressions de l'erreur de méthode et de l'erreur de calcul

En supposant φ dérivable et à dérivées continues sur $[x-h,\ x+h]$ jusqu'à l'ordre trois, et en développant les termes du numérateur de (4) on obtient:

$$e_m = \frac{-h^2 \varphi'''(\zeta)}{6}, \qquad \zeta \in (x-h, x+h) \tag{10}$$

Quant à l'erreur de calcul e_c, en supposant qu'elle n'est due qu'à l'image machine Φ de φ, nous pouvons en donner une approximation. En effet l'erreur relative sur Φ au point x est donnée par $\varepsilon(x)$ telle que

$$\Phi(x) - \varphi(x) = \varepsilon(x)\,\varphi(x) \tag{11}$$

Si on introduit l'hypothèse qu'il existe P, $0 < P \ll 1$ tel que pour tout $x \in R$ (ensemble de définition de φ), l'on)ait $|\varepsilon(x)| \leqq P$ et si on néglige dans ce calcul d'erreur les termes en εh, alors à partir de (11) et (7) on obtient:

$$e_c = \frac{\Phi(x)}{2h}(\varepsilon_1 - \varepsilon_2) \tag{12}$$

où $\varepsilon_1 = \varepsilon(x-h)$, $\varepsilon_2 = \varepsilon(x+h)$, ε_1 et $\varepsilon_2 \in [-P, P]$.

Dumontet montre, à partir de (12) que $|e_c| \uparrow$ si $h \downarrow$. Il en résulte donc $\varphi'(x) = D(H) + e_m + e_c$ où $|e_m| \downarrow$ et $|e_c| \uparrow$ si $h \downarrow$.

La seule méthode valable dans ce genre d'approximation est donc celle qui choisit un pas de dérivation *h tel que la composition de* e_m et e_c *soit minimale.*

Plusieurs méthodes ont été proposées pour cela, nous signalerons la méthode de Dumontet [6] et la méthode de Stepleman et Winarsky [11].

Il semble cependant que l'inconvénient majeur de l'une comme de l'autre reste le nombre élevé d'évaluations de la fonction à dériver (en moyenne 15 appels pour le calcul d'une dérivée par la première méthode et environ 7 par la deuxième méthode).

Première méthode

La méthode de Dumontet propose de calculer h^* tel que la moyenne de $|e_m + e_c|$ soit minimale. Ce calcul est donné dans [6] et on obtient:

$$h^* = \sqrt[3]{\frac{1.67\,P\,|\Phi(x)|}{|\varphi'''(x)|}}$$

La formule (13) fait intervenir $\varphi'''(x)$ par sa racine cubique. En réalité on se contentera d'une mauvaise approximation de $\varphi'''(x)$ pour avoir une bonne approximation de $\varphi'(x)$. Nous utiliserons l'approximation suivante de $\varphi'''(x)$:

$$\Phi'''(x) = \frac{\Phi(x+2k) - \Phi(x-2k) - 2(\Phi(x+k) - \Phi(x-k))}{2k^3} \tag{14}$$

La formule (14) donne mathématiquement une erreur $\varepsilon(k)$ tendant vers zéro lorsque $k \to 0$. Aussi k sera t-il choisi de sorte que e_m dans cette approximation soit négligeable quitte à avoir volontairement e_c élevée, sans toutefois dépasser une certaine limite (c'est-à-dire que l'erreur globale doit être majorée).

Choix d'une valeur correcte pour k

En utilisant les formules (11) et (14) et en donnant à $\varepsilon(x)$ les valeurs extrêmes $-P$ et $+P$, on peut déterminer les expressions des limites sup et inf de $f''(x)$ en fonction de P, K et F. Comme $P \ll 1$ on peut écrire $\Phi'''(x) = (\varphi'''_{\text{sup}}(x) + \varphi'''_{\text{inf}}(x))/2$. Notons L le rapport $\varphi'''_{\text{sup}}(x)/\varphi'''_{\text{inf}}(x)$.

- e_m trop grande se traduit alors par L très proche de 1
- e_c trop grande se traduit par L trop éloigné de 1.

Il est donc nécessaire de choisir k de sorte que L soit à une distance raisonnable de l'unité. Ce qui s'écrira:

$$L \in [L_1, L_2] \qquad \text{et} \qquad L \notin [L_3, L_4] \tag{15}$$

Il a été montré dans [6] qu'il existe des valeurs de L_1, L_2, L_3 et L_4 universelles, ce sont respectivement 1/15, 15, 1/2 et 2. Une dichotomie dans [KINF, KSUP]

donne la valeur k qui satisfait la condition (15); on commence dans l'intervalle $[X^* 2^{-m+1}, X^* 2^{m-1}]$ avec $k = \sqrt{\text{KINF}^* \text{KSUP}}$.

C'est de cette recherche de k que découle le grand nombre d'évaluations de Φ, gros inconvénient de la méthode de Dumontet.

Deuxième méthode

Quant à la méthode de Stepleman et Winarsky, elle propose d'éviter l'évaluation des erreurs de méthode (troncature) et de calcul (arrondi). Elle est basée sur le principe suivant:

— *si on choisit une suite* $\{h_i\}$ *strictement monotone tendant vers* 0 (h_0 *suffisamment petit*), *alors pour toute fonction* f *différentiable, la suite* $\{d(h_i)\}$ *tend monotonement vers* $\varphi'(x)$ *quand* $h_i \to 0$.

Critère d'arrêt dans l'approximation de $\varphi'(x)$

En pratique on ne peut pas atteindre la limite de la suite $\{h_i\}$ (sauf si elle est stationnaire). Il faudra donc décider quand arrêter de générer des éléments $D(H_i)$ et considérer qu'on a obtenu une bonne approximation de $\varphi'(x)$ par cette méthode de Stepleman et Winarsky.

Nous allons définir une relation pouvant nous servir de critère d'arrêt, en posant des hypothèses plus fortes. On pourra par exemple supposer que la fonction définie par

$$\Psi(t) = f(x+t) - f(x-t)$$

est de classe C^3 sur $I = [0, \varepsilon]$, ε arbitraire, avec Ψ'' et Ψ''' de signes constants sur I. Alors pour le choix (h_i) vérifiant $h_{i+1} = h_i/\beta$ pour $\beta > 1$, s'il n'y avait pas d'erreur d'arrondi, on devrait avoir $d(h_i) - d(h_{i+1})$ de l'ordre de $f''(x)(h_i^2 - h_{i+1}^2)/6$, et donc:

$$|d(h_i) - d(h_{i+1})| < |d(h_{i-1}) - d(h_i)| \quad \text{pour } h_i \text{ petit.}$$

En fait nous utilisons dans notre programmation la relation suivante:

$$|d(h_i) - d(h_{i+1})| \leqq |d(h_{i-1} - d(h_i)| \neq 0, \qquad i \geqq N \tag{16}$$

En pratique la relation (16) ne peut être indéfiniment vérifiée vue la propagation des erreurs d'arrondi due à la précision finie de la machine.

Sur ordinateur il existe un ensemble non vide d'éléments H_i pour lesquels $D(H_i) = 0$, dans ce cas toute signification est perdue. Mais parmi les H_i n'appartenant pas à cet ensemble il existe forcément un H_j pour lequel (16)

n'est pas vérifiée (c'est-à-dire $|D(H_i)-D(H_{i+1})|>|D(H_i)-D(H_{i-1})|$, le cas où les deux membres sont nuls est testé à part). On approxime alors $\varphi'(x)$ par $D(H_{j-1})$.

4. APPROXIMATION DE LA MATRICE HESSIENNE DANS LA MÉTHODE DE NEWTON

Cette approximation du hessien a été faite dans le cadre du problème d'optimisation non contrainte grâce à un algorithme donné dans Salhi [10] (les lignes générales ayant déjà été citées au début du paragraphe 2).

Le point crucial dans cette approximation étant le choix du pas H, nous avons adopté la méthode de Stepleman et Winarsky en y incluant quelques modifications d'ordre numérique.

La construction de la suite $\{H_i\}$ est effectuée suivant l'algorithme ci-dessous donné et justifié dans [10] et [11], la dérivation s'effectue sur le gradient de la fonction à optimiser. Si $f: R^n \to R$ alors le gradient est défini de R^n dans R^n. Pour calculer l'élément de rang (i,j) du hessien il faudra donc déterminer le pas H tel que la valeur $D(H)$ calculée par (17) soit une bonne approximation:

$$D(H)=(G_i(X \oplus H \circledast e_j) \ominus G_i(X \ominus H \circledast e_j)) \oslash 2 \circledast H \qquad (17)$$

G_i est la i-ième composante du gradient G

e_j est le j-ième vecteur de la base canonique de R^n

X est le vecteur de R^n où s'effectue le calcul de la dérivée.

L'algorithme de recherche de pas H est appliqué à tout $i=1,\ldots,n$ et $j=1,\ldots,n$.

Vu les conditions posées sur la fonction le hessien est symétrique. On réduit alors le nombre d'éléments à approximer de $n*n$ à $n*(n-1)/2$. L'algorithme décrit la recherche du pas H pour *un élément* du hessien en un point X donné.

Algorithme 1 de recherche de pas pour l'élément de rang (i,j)

Étape 1 :

$$H_0 = \beta * p^{1/3} * X_j$$

où

β est un réel choisi supérieur à 1

P est égal à 2^{-m}, où m représente le nombre de bits de la mantisse pour la machine utilisée.

X_j est la composante du point où s'effectue la dérivation

Étape 2 :

Si $G_i(X+H_0*e_j) * G_i(X-H_0*e_j) \leqq 0$ aller à l'étape 6.

Étape 3 :

Calculer : $(H_0) = 12*H_0*D(H_0)/G_i(X)|$, $N(H_0) = -\log(H_0)$.

Étape 4 :

Si $0 \leqq N(H_0) \leqq \mathrm{Log}(P^{-1/3}/\beta)$ aller à l'étape 6.

Étape 5 :

Chercher une nouvelle valeur de H_0 par dichotomie** et reprendre à l'étape 3.

Étape 6 :

Calculer $H_{k+1} = H_k/\beta$,

Étape 7 :

Si $|D(H_{k+1}) - D(H_k)| \leqq |D(H_{k-1}) - D(H_k)| \neq 0$ aller à l'étape 6.

Étape 8 :

Poser $H^* = H_k$ et $\partial G_i(x)/\partial x_j = D(H^*)$; FIN.

Dichotomie**

Soit $H_0 = \beta * P^{1/3} * X$ l'approximation initiale ne vérifiant pas

$$0 \leqq N(H_0) \leqq \mathrm{Log}(P^{-1/3}/\beta) \quad (18).$$

Selon la partie non vérifiée de la double inégalité (18) on procède comme suit :

– Si $N(H_0) > \mathrm{Log}(P^{-1/3}/\beta)$, (18.1) est non vérifiée, remplacer H_0 par H_0/β, calculer $N(H_0)$ pour cette nouvelle valeur de H_0 et répéter le processus tant que (18.1) n'est pas vérifiée.

– Si $N(H_0)<0$ (18.2) est non vérifiée, poser

$$H_0^1=H_0/\beta,\ H_0^2=(H_0+H_0^1)/2 \qquad \text{et} \qquad H_0=H_0^2.$$

Calculer $N(H_0)$ pour cette nouvelle valeur de H_0 et répéter le processus tant que (18) n'est pas vérifiée.

Remarque: Au cours de l'une quelconque des étapes il peut arriver que $D(H_k)=0$. Il y a alors plusieurs possibilités:

1. le pas H_k est non significatif devant X_j d'où

$$X+H_k{}^*e_j=X-H_k{}^*e_j$$

2. la valeur $D(H_k)$ est non significative soit parce que la i-ième composante du gradient ne dépend pas de X_j ($D(H_k)$ étant alors un zéro mathématique), soit parce que la différence entre deux valeurs voisines du gradient ne représente que l'effet cumulé des erreurs d'arrondi.

On ne peut pas différencier ces divers cas sans l'utilisation d'une méthode appropriée de calcul de précision (ceci a été fait dans le cadre de notre étude; on pourra consulter à ce sujet Salhi [10]), on se contentera ici d'arrêter la recherche du pas optimal et de considérer que le pas précédent est correct c'est-à-dire:

$$H^*=H_{k-1}, \qquad k=1,2,\ldots \quad (\text{si } k=0,\ H^*=H_0)$$

et

$$\partial G_i(x)/\partial x_j=D(H^*)$$

De même $|D(H_k)-D(H_{k+1})|$ peut ne représenter que le cumul des erreurs, ainsi les calculs effectués à partir de là seront non significatifs. Il est donc évidemment inutile de continuer à générer des éléments de la suite de pas de dérivation car on aboutit à un bouclage (d'où l'arrêt des itérations de recherche de pas si dans (16) on a les deux membres nuls).

5. MISE EN OEUVRE ET RÉSULTATS OBTENUS SUR CRAY-1

Les résultats suivants sont obtenus sur Cray-1 en appliquant l'algorithme 1 de recherche de pas exposé au paragraphe 4. Nous retrouvons dans le tableau II les fonctions données au tableau I, ce qui permet de mener une comparaison

entre:

— l'utilisation d'un pas H fixe pour approximer la matrice hessienne dans la méthode de Newton, d'une part, et

— l'utilisation de l'algorithme 1 donné au paragraphe 4 d'autre part.

Comparons, maintenant, le nombre d'évaluations du gradient exigé par l'utilisation systématique de l'algorithme 2 (recherche de pas à chaque approximation de la matrice hessienne) et ceux donnés au tableau numéro 1 par l'algorithme 1 (utilisant trois pas de dérivation *fixés* par l'utilisateur).

Notons *Ngs* le nombre d'évaluations du gradient exigé par l'utilisation de l'algorithme 2 et *Ng* celui exigé par l'algorithme 1.

Stratégie de calcul de *Ng*

Les trois pas utilisés dans notre expérience numérique sont

$$H_1 = 10^{-2}, \qquad H_2 = 10^{-7} \qquad \text{et} \qquad H_3 = 10^{-12}.$$

Nous avons constaté qu'en général le bon pas se situe autour du pas H_2. Il est alors intéressant de procéder comme suit dans l'approximation du hessien dans la méthode d'optimisation utilisée (décrite dèjà au paragraphe 2):

(*a*) En première résolution du problème utiliser le pas H_2 pour approximer le hessien. Soit $Ng1$ le nombre d'évaluations du gradient effectuées et $FX1$ la valeur de la fonction à l'arrêt.

(*b*) En deuxième résolution utiliser le pas H_1. Soit $Ng2$ le nombre d'appels du gradient et $FX2$ la valeur de la fonction à l'itération courante.

Tant que l'arrêt n'a pas lieu tester: $Ng2 > Ng1$

— *Si oui* vérifier s'il y a une amélioration par rapport à la première résolution, c'est-à-dire: $FX2 < FX1$.

— *Si oui* continuer les itérations l'arrêt aura lieu dans peu d'itérations (cas rare).

— *Si non* stopper les itérations.

— *Si non* continuer.

(*c*) En troisième résolution utiliser le pas H_3. Soit $Ng3$ le nombre d'appels du gradient et $FX3$ la valeur de la fonction à l'itération courante. Procéder comme pour le pas H_1 mais en comparant $Ng3$ au plus petit entre $Ng1$ et $Ng2$ et $FX3$ à la valeur de la fonction correspondant à celui choisi entre $Ng1$ et $Ng2$.

TABLEAU II

– La première colonne donne le numéro du problème testé.

– La deuxième colonne la solution et la valeur de la fonction trouvée.

– La troisième colonne le nombre d'itérations (iter) et le nombre d'évaluations du gradient (*Ngs*).

– La quatrième colonne la solution mathématique du problème testé.

fct	solution et fonction	iter et Ngs	x* et fx*
1	X1 = 0.999999999998337E+00 X2 = 0.999999999994902E+00 FX = 0.382596276854742E-23	iter = 28 Ngs = 582	X1 = 1.0 X2 = 1.0 FX = 0.0
2	X1 = 0.299999999999972E+01 X2 = 0.499999999999936E+00 FX = 0.000000000000000E+00	iter = 7 Ngs = 164	X1 = 3.0 X2 = 0.5 FX = 0.0
3	X1 = 0.400000000002885E+01 X2 = 0.899999999997431E+01 FX = -0.182000000000008E+02	iter = 3 Ngs = 51	X1 = 4.0 X2 = 9.0 FX =-18.2
4	X1 = -0.137571365127149E-08 X2 = 0.138431760514390E-08 X3 = 0.999999999725620E+00 FX = 0.755052693304981E-18	iter= 100 * Ngs = 3727	X1 = 0.0 X2 = 0.0 X3 = 1.0 FX = 0.0
5	X1 = 0.100000000000000E+01 X2 = 0.100000000000000E+01 X3 = 0.100000000000000E+01 X4 = 0.100000000000000E+01 FX = 0.000000000000000E+00	iter = 15 Ngs = 888	X1 = 1.0 X2 = 1.0 X3 = 1.0 X4 = 1.0 FX = 0.0
6	X1 = 0.241308750242140E-04 X2 = -0.241308750238363E-05 X3 = 0.324477658905761E-05 X4 = 0.324477658910965E-05 FX = 0.000000000000000E+00	iter = 30 Ngs = 1709	X1 = 0.0 X2 = 0.0 X3 = 0.0 X4 = 0.0 FX = 0.0
7	X1 = 0.100000000000001E+01 X2 = 0.100000000000003E+01 FX = 0.000000000000000E+00	iter = 6 Ngs = 165	X1 = 1.0 X2 = 1.0 FX = 0.0
8	X1 = 0.100000000000001E+01 X2 = -0.365701392779361E-15 FX = 0.000000000000000E+00	iter = 7 Ngs = 172	X1 = 1.0 X2 = 0.0 FX = 0.0
9	X1 = 0.878879400837592E-24 X2 = -0.242338070083895E-25 X3 = 0.484676140167790E-26 FX = 0.000000000000000E+00	iter = 2 Ngs = 271	X1 = 0.0 X2 = 0.0 X3 = 0.0 FX = 0.0

TABLEAU II (*suite*)

10	X1 = 0.951342894102862E-03 X2 = 0.100096130938177E+01 X3 = 0.100000224833683E+01 X4 = 0.100000000000000E+01 FX = 0.000000000000000E+00	iter = 28 Ngs = 2811	X1 = 0.0 X2 = 1.0 X3 = 1.0 X4 = 1.0 FX = 0.0
11	X1 = 0.780601717342925E-39 X2 = 0.780601717342925E-39 X3 = 0.780601717342925E-39 X4 = 0.780601717342925E-39 X5 = 0.780601717342925E-39 X6 = 0.780601717342925E-39 X7 = 0.780601717342925E-39 X8 = 0.780601717342925E-39 X9 = 0.734683969263930E-39 X10= 0.734683969263930E-39 X11= 0.780601717342925E-39 X12= 0.780601717342925E-39 X13= 0.826519465421921E-39 X14= 0.826519465421921E-39 X15= 0.780601717342925E-39 X16= 0.780601717342925E-39 X17= -0.806097978223278E-30 X18= 0.704709803713442E-31 X19= 0.780601717342925E-39 X20= 0.780601717342925E-39 FX = 0.000000000000000E+00	iter = 3 Ngs = 3988	X1 = 0.0 X2 = 0.0 X3 = 0.0 X4 = 0.0 X5 = 0.0 X6 = 0.0 X7 = 0.0 X8 = 0.0 X9 = 0.0 X10= 0.0 X11= 0.0 X12= 0.0 X13= 0.0 X14= 0.0 X15= 0.0 X16= 0.0 X17= 0.0 X18= 0.0 X19= 0.0 X20= 0.0 FX = 0.0
12	X1 = 0.333333333333332E+00 X2 = 0.366666666666664E+00 X3 = 0.999999999999996E+00 X4 = 0.133333333333333E+01 X5 = 0.166666666666666E+01 X6 = 0.199999999999999E+01 X7 = 0.233333333333331E+01 X8 = 0.266666666666666E+01 X9 = 0.299999999999999E+01 X10= 0.333333333333331E+01 X11= 0.366666666666664E+01 X12= 0.399999999999999E+01 X13= 0.433333333333331E+01 X14= 0.466666666666663E+01 X15= 0.499999999999997E+01 X16= 0.533333333333331E+01 X17= 0.566666666666663E+01 X18= 0.599999999999997E+01 X19= 0.633333333333329E+01 X20= 0.666666666666663E+01 FX = 0.000000000000000E+00	iter = 4 Ngs = 4407	X1 = 1/3 X2 = 2/3 X3 = 1.0 X4 = 4/3 X5 = 5/3 X6 = 2.0 X7 = 7/3 X8 = 8/3 X9 = 3.0 X10= 10/3 X11= 11/3 X12= 4.0 X13= 13/3 X14= 14/3 X15= 5.0 X16= 16/3 X17= 17/3 X18= 6.0 X19= 19/3 X20= 20/3 FX = 0.0

* 100 est le nombre maximal d'itérations autorisées.

Le nombre total d'évaluations du gradient Ng est égal alors à $Ng1+Ng2+Ng3$. Le tableau I montre qu'en général Ng prend une valeur du type 3* $Ng1$ ou $Ng1+2*$ $Ng2$ ou $Ng1+Ng2+Ng3$.

TABLEAU III

Fct	Nombre d'appels du gradient	
	Ngs (algorithme 2)	Ng (algorithme 1)
1	582	480 = 3 * 160
2	164	120 = 3 * 40
3	51	45 = 3 * 15
4	3727	399 = 3 * 133
5	888	468 = 126 + 135 + 207
6	1709	837 = 3 * 279
7	165	105 = 3 * 35
8	172	140 = 45 + 50 + 45
9	271	63 = 3 * 21
10	2811	810 = 3 * 270
11	3988	492 = 123 + 164 + 205
12	4407	820 = 164 + 246 + 410

Nous comparons alors, au tableau III, Ngs à l'une des trois valeurs possibles de Ng (le choix étant régi par la stratégie décrite ci-dessus) pour chacune des fonctions numérotées de 1 à 12.

Interprétation

On constate que le nombre d'appels du gradient nécessaire lorsqu'on utilise l'algorithme 2 est souvent supérieur à la somme de ceux exigés par trois exécutions consécutives du programme avec des pas fixés *a priori* (algorithme 1). Cette élévation du nombre d'appels du gradient constitue un gros inconvénient de l'utilisation de l'algorithme 2. De même nous remarquons que pour la fonction 4 l'algorithme 2 donne 100 itérations contre $3*18=54$

pour l'algorithme 1, où 100 est le nombre maximal d'itérations autorisées. La précision de l'algorithme 1 est alors dans ce cas bien meilleure.

Il se pose alors un problème de choix entre:

– Un algorithme dépendant d'une valeur H fixée au départ et inconnue *a priori*, alors qu'elle est capitale dans le bon déroulement et la convergence de celui-ci, d'une part, et

– Un algorithme dont la convergence est plus sûre mais exigeant un nombre d'évaluations du gradient trop élevé d'autre part.

Nous proposons alors de combiner les deux algorithmes de façon à bénéficier de la rapidité de l'un (pour un H fixé correct) et de l'automatisme du second (recherche d'un pas optimal dépendant des caractéristiques de la fonction traitée).

Nous désignerons ce nouvel algorithme par « algorithme 3 ». Il comporte 3 étapes principales:

– Les deux premières déterminent le pas de dérivation.

– La troisième propose le calcul des éléments du hessien avec les pas trouvés puis le calcul de la direction de recherche dans le problème d'optimisation par la méthode de Newton. Si la direction ainsi trouvée ne fournit pas une diminution de la valeur de la fonction objective nous proposons à la sous étape 3.1 de reprendre la recherche des pas de dérivation pour améliorer cette direction. Si la tentative échoue nous proposons en dernier recours de changer de direction de recherche et suggérons le modèle suivant que nous désignons par « Modèle** ».

Modèle de changement de direction**

Ce modèle de choix est donné dans Salhi [10]. Il propose:

– l'utilisation de la direction de Newton calculée à partir du hessien tant que cette direction est direction de descente et qu'elle permet une amélioration (diminution) de la valeur de la fonction.

– Si la direction de Newton n'est pas direction de descente ou si elle correspond à un piétinement sur une valeur de la fonction, changer la direction de recherche z^k en utilisant la direction $-g^k$, g^k étant le gradient de la fonction.

– si la direction $-g^k$ ne permet pas une amélioration de la valeur de la fonction, tester l'existence d'arêtes sur l'hypersurface représentant la fonction. Dans le cas positif utiliser la direction opposée de la bissectrice de deux gradients consécutifs pour contourner cette arête.

6. ALGORITHME 2 DE RECHERCHE DE PAS, MISE EN OEUVRE ET RÉSULTATS

Cet algorithme est basé sur le principe suivant:

Étape 1:

Dérouler l'algorithme 1 de recherche de pas à l'itération k (on commencera à $k=0$) c'est-à-dire en x^k, en stockant les valeurs des pas H_{ij} correspondant respectivement à $\partial^2 f(x^k)/\partial x_i \partial x_i \partial x_j$, $i=1, \ldots, j$ et $j=1, \ldots, n$.

Étape 2:

Calculer les pas $H_j=(1/n) \sum_{i=1}^{n} H_{ij}$, $j=1, \ldots, n$.

Étape 3:

Utiliser chaque pas H_j pour calculer une approximation de $\partial^2 f(x^k)/\partial x_i \partial x_j$, $i=1, \ldots, j$ et ceci pendant 10 itérations.

3.1. Si au cours de ces 10 itérations, un problème est détecté notamment la non-variation de la valeur de la fonction, on tentera d'améliorer la direction de déplacement en calculant à nouveau la matrice hessienne grâce à l'algorithme 1 au point x^{k+l}, $0<l<10$.

Il est clair qu'un piétinement peut survenir même si le hessien est exact. Aussi, si la tentative d'amélioration en x^{k+l} ne donne pas une meilleure valeur de la fonction. il est nécessaire de changer la direction de recherche en employant une autre méthode que celles utilisant le hessien (nous proposons le modèle** précédent). Par contre si la tentative est satisfaisante on retourne à l'étape 2 avec $k=k+l$.

3.2. Si aucun problème n'est rencontré, nous continuons avec les mêmes pas H_j, $j=1, \ldots, n$ jusqu'à la dixième itération. En x^{k+10}, la recherche des pas H_{ij} est reprise à l'étape avec $k=k+10$.

Pour toute justification des étapes 2 et 3, le lecteur pourra se reporter à [10] où sont exposées les différentes étapes de recherches ayant conduit à l'algorithme 3.

TABLEAU IV

Ngm désigne le nombre d'évaluations du gradient effectuées en utilisant l'algorithme 3. Ce nombre contient aussi les appels effectués pour les tests d'arrêts en dehors du calcul du hessien.

fct	solution et fonction	iter et Ngm	x* et fx*
1	X1 = 0.999999999998458E+00 X2 = 0.999999999995271E+00 FX = 0.329751431447073E-23	iter = 29 Ngm = 191	X1 = 1.0 X2 = 1.0 FX = 0.0
2	X1 = 0.299999999999969E+01 X2 = 0.499999999999931E+00 FX = 0.169636649058726E-25	iter = 7 Ngm = 48	X1 = 3.0 X2 = 0.5 FX = 0.0
3	X1 = 0.400000000002885E+01 X2 = 0.899999999997431E+01 FX = -0.182000000000008E+02	iter = 3 Ngm = 27	X1 = 4.0 X2 = 9.0 FX =-18.2
4	X1 = 0.366611006903038E-14 X2 = -0.370637894178514E-15 X3 = 0.100000000000000E+01 FX = 0.000000000000000E+00	iter = 17 Ngm = 180	X1 = 0.0 X2 = 0.0 X3 = 1.0 FX = 0.0
5	X1 = 0.999999999999986E+00 X2 = 0.999999999999968E+00 X3 = 0.100000000000003E+01 X4 = 0.100000000000006E+01 FX = 0.640681272784323E-26	iter = 18 Ngm = 254	X1 = 1.0 X2 = 1.0 X3 = 1.0 42 = 1.0 FX = 0.0
6	X1 = 0.165688651229021E-04 X2 = -0.165688651229513E-05 X3 = -0.426321724979753E-05 X4 = -0.426321725001925E-05 FX = 0.188557566111885E-17	iter = 30 Ngm = 415	X1 = 0.0 X2 = 0.0 X3 = 0.0 X4 = 0.0 FX = 0.0
7	X1 = 0.100000000000001E+01 X2 = 0.100000000000001E+01 FX = 0.631088724176809E-28	iter = 6 Ngm = 47	X1 = 1.0 X2 = 1.0 FX = 0.0
8	X1 = 0.999999999998685E+00 X2 = -0.227798538182764E-11 FX = -0.106581410364015E-13	iter = 9 Ngm = 63	X1 = 1.0 X2 = 0.0 FX = 0.0
9	X1 = 0.516987882845642E-25 X2 = 0.258493941422821E-25 X3 = 0.145402842050337E-25 FX = 0.572921732290270E-50	iter = 2 Ngm = 39	X1 = 0.0 X2 = 0.0 X3 = 0.0 FX = 0.0
10	X1 = 0.951616663837120E-03 X2 = 0.100096158353527E+01 X3 = 0.100000252277260E+01 X4 = 0.100000000000000E+01 FX = 0.778256384712770E-16	iter = 28 Ngm = 567	X1 = 0.0 X2 = 1.0 X3 = 1.0 X4 = 1.0 FX = 0.0

100 est le nombre maximal d'itérations autorisées.

7. EXPÉRIENCE NUMÉRIQUE ET RÉSULTATS

Le logiciel obtenu a été mis-en-œuvre sur Cray-1.

Nous donnons au tableau numéro 4 les résultats correspondant fonctions testées et présentées en annexe. Les fonctions 11 et 12 n'ont été utilisées que pour montrer l'augmentation du nombre d'appels du gradient en passant de l'algorithme 1 à l'algorithme 2. Dans un cas concret il serait plus profitable de tenir compte du fait que les hessiens de ces fonctions sont constants, d'où l'inutilité de rechercher le pas de dérivation, aussi ne seront-elles pas reprises dans la suite de l'étude.

TABLEAU V

Fct	Diminution réalisée par l'heuristique (en %)		itérations	
	appels du gradient	temps d'exécution	algo2	algo3
1	67 %	56,5 %	28	29
2	70,7 %	61 %	7	7
3	95 %	94 %	3	3
4	46 %	31 %	100 *	17
5	71,4 %	61 %	15	18
6	76 %	69 %	30	30
7	71,5 %	68,5 %	6	6
8	64 %	51 %	7	6
9	85,6 %	83 %	2	2
10	80 %	78 %	28	28

* 100 est le nombre maximal d'itérations autorisées.

Les résultats ci-dessus montrent le bien-fondé de l'algorithme 3. En effet il en ressort que pour un nombre d'itérations fixe, les nombres d'évaluations du gradient diminuent considérablement avec l'emploi de l'algorithme 3 au lieu de l'algorithme 2 ou l'algorithme 1. Afin de mieux visualiser cette diminution, nous présentons dans le tableau V les pourcentages de gain en appels du gradient et en temps calcul réalisé par l'algorithme 3 par rapport

à l'algorithme 2 ainsi que les nombres d'itérations exigés par ces deux algorithmes.

Ces pourcentages ont été obtenus en appliquant la formule suivante:

$$P\% = N/D$$

où N est la valeur trouvée par l'algorithme 3 et D celle trouvée par l'algorithme 2.

CONCLUSION

Le modèle algorithmique présenté dans [10] se veut répondre au souci d'efficacité, de stabilité numérique et d'adaptation aux nouvelles possibilités de l'informatique, tout en tenant compte des considérations de coût lors de la résolution d'un problème d'optimisation.

Dans cet article, nous nous sommes intéressé à l'efficacité du modèle proposé dans [10]. Nous avons présenté le problème considéré en soulevant les difficultés majeures de l'approximation sur ordinateur et souligné les contraintes à respecter pour obtenir la meilleure approximation possible au sens informatique.

Après avoir mis en évidence le point crucial en dérivation numérique, à savoir le choix du pas, nous avons proposé une méthode originale de recherche du pas optimal tenant compte de l'aspect mathématique et informatique de l'approximation de la matrice hessienne:

Un pas H sera dit optimal s'il correspond à une erreur globale (*erreur de troncature + erreur d'arrondi*) minimale. Cette méthode permet ainsi:

— le calcul des dérivées secondes sans avoir recours à leurs expressions analytiques

— l'approximation au mieux du hessien en déterminant le meilleur pas de dérivation, c'est-à-dire celui minimisant les erreurs

— la réalisation de cette approximation au moindre coût et ce en minimisant le nombre d'appels du gradient.

Nous terminons l'exposé en présentant les résultats obtenus par la méthode traitée. Plusieurs autres améliorations d'ordre numérique ont été effectuées sur l'heuristique présentée en algorithme 3.

Elle a été en premier lieu stabilisée grâce à une analyse de la propagation des erreurs de calculs au cours du déroulement sur ordinateur. Cette stabilisation a

permis notamment de connaître exactement la précision des résultats obtenus et d'arrêter les calculs au moment utile.

De plus le logiciel d'optimisation obtenu ayant été mis en œuvre sur ordinateur vectoriel, il serait illogique de ne pas tenir compte de l'architecture pipe-line de l'ordinateur. Ceci a été fait grâce à la vectorisation de tout le logiciel d'optimisation non contrainte proposé dans Salhi [10] et auquel le lecteur pourra se reporter pour plus d'information sur les points essentiels suivants :

– Dérivation numérique

– Optimisation non contrainte de fonctions quelconques avec détection d'arêtes

– Stabilisation numérique par analyse statistique des erreurs d'arrondi

– Vectorisation sur Cray-1.

Nous espérons avoir répondu en partie au problème posé et avoir contribué à alléger l'utilisation pratique des méthodes à dérivées secondes (caractérisées par leur convergence rapide en général) en optimisation.

ANNEXE

Fonction numéro 1 : *White et Holst*

$$f(x)=100\,(x_2-x_1^3)^2+(1-x_1)^2$$

avec $x_0=(-1.2,\ 1.)$; $x^*=(1.,\ 1.)$: $f(x^*)=0$.

Fonction numéro 2 : *Beale*

$$f(x)=(1.5-x_1\,(1-x_2))^2+(2.25-x_1\,(1-x_2)^2)^2+(2.625-x_1\,(1-x_2)^3)^2$$

avec $x_0=(1.,\ 0.8)$ ou $(2.,\ 0.2)$; $x^*=(3.,\ 0.5)$: $f(x^*)=0$.

Fonction numéro 3 : *Zangwill*

$$f(x)=(1/15)\,(16\,x_1^2+16\,x_2^2-8\,x_1\,x_2-56\,x_1-256\,x_2+991)$$

avec $x_0=(3.,8.)$; $x^*=(4.,9.)$; $f(x^*)=-18.2$.

Fonction numéro 4: *Engvall*

$$f(x)=\sum_{i=1}^{5} f_i(x)^2$$

où

$$f_1(x)=x_1^2+x_2^2+x_3^2-1, \qquad f_2(x)=x_1^2+x_2^2+(x_3-2)^2-1$$
$$f_3(x)=x_1+x_2+x_3-1, \qquad f_4(x)=x_1+x_2-x_3+1$$
$$f_5(x)=x_1^2+3x_2^2+(5x_3-x_1+1)^2-36$$

avec $x_0=(1.,2.,0.)$; $x^*=(0.,0.,1.)$; $f(x^*)=0.$

Fonction numéro 5: *Wood-Colville*

$$f(x)=100(x_2-x_1^2)^2+(1-x_1)^2+90(x_4-x_3^2)^2+(1-x_3)^2$$
$$+10.1[(x_2-1)^2+(x_4-1)^2]+19.8(x_2-1)(x_4-1)$$

avec $x_0=(3.,\ 1.,\ 3.,\ 1.)$; $x^*=(1.,\ 1.,\ 1.,\ 1.)$; $f(x^*)=0.$

Fonction numéro 6: *Powell*

$$f(x)=(x_1+10x_2)^2+5(x_3-x_4)^2+(x_2-2x_3)^4+10(x_1-x_4)^4$$

avec $x_0=(3.,\ 1.,\ 0.,\ -1.)$; $x^*=(0.,\ 0.,\ 0.,\ 0.)$; $f(x^*)=0.$

Fonction numéro 7: *Box*

$$f(x)=\sum_{i=1}^{10}[\exp(-x_1t_i)-\exp(-x_2t_i)-\exp(-t_i)+\exp(-10t_i)]^2$$

avec $t_i=0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1.0$

$$x_0=(4.,6.);\ x^*=(1.,10.);\ f(x^*)=0.$$

Fonction numéro 8: *Engvall*

$$f(x)=x_1^4+x_2^4+2x_1^2x_2-4x_1+3$$

avec $x_0=(0.5,\ 2.0)$; $x^*=(1.0,\ 0.)$; $f(x^*)=0.$

Fonction numéro 9: *Zangwill*

$$f(x)=(x_1-x_2+x_3)^2+(-x_1+x_2+x_3)^2+(x_1+x_2-x_3)^2$$

avec $x_0=(100.,\ -1.,\ 2.5)$; $x^*=(0.,\ 0.,\ 0.)$; $f(x^*)=0.$

Fonction numéro 10: *Cragg et Levy*

$$f(x)=[\exp(x_1)-x_2]^4+100(x_2-x_3)^6+\tan^4(x_3-x_4)+x_1^8+(x_4-1)^2$$

avec $x_0=(1.,\ 2.,\ 2.,\ 2.)$; $x^*=(0.,\ 1.,\ 1.,\ 1.)$; $f(x^*)=0.$

Fonction numéro 11:

$$f(x)=\sum_{i=1}^{20} i!\,x_i^2$$

avec $x_0=(-1.,\ -1.,\ \ldots,\ -1.)$; $x^*=(0.,\ 0.,\ \ldots,\ 0)$; $f(x^*)=0.$

Fonction numéro 12:

$$f(x)=\sum_{i=1}^{20} i!\,(x_i-i/3)^2$$

avec $x_0=(-1.,\ -1.,\ldots,-1.)$; $x^*=(1/3,\ 2/3,\ \ldots,20/3)$; $f(x^*)=0.$

BIBLIOGRAPHIE

1. J. Abadie, *Integer and Nonlinear Programming*, North-Holland, Amsterdam, 1970.
2. J. Abadie, *Advances in Nonlinear Programming*, Operational research 78, Proceeding of Eigth IFORS International Conference of Operational Research, K. B. Haley éd., 1978, p. 900-990.
3. J. Abadie, *Programmation mathématique*, Cours de D.E.A., Université Pierre-et-Marie-Curie, Paris-VI, 1983.
4. E. De Drouas, *Vectoriser sur le Cray*-1, Bulletin de la direction des études et recherches, Serie C, E.D.F.
5. F. Dekhli, *Étude et réalisation de logiciels d'optimisation contrainte avec analyse des erreurs de calculs et de données*, Thèse, Université Pierre-et-Marie-Curie, Paris-VI, 1985.
6. J. Dumontet, *Algorithme de dérivation numérique. Étude théorique et mise en œuvre sur ordinateur*, Thèse, Paris-VI, 1973.
7. H. Hasni, *Logiciels vectoriels d'optimisation de problèmes non contraints de grande taille et calcul de la précision*, Thèse, Université Pierre-et-Marie-Curie, Paris-VI, 1986.
8. M. La Porte et J. Vignes, *Algorithmes numériques. Analyse et mise en œuvre. Arithmétique des ordonateurs. Systèmes linéaires*, t. 1, Technip. éd., 1974.

9. Y. PERROT, *Modélisation et simulation d'une base de connaissances à exécution segmentée, parallèle et répartie*, Doctorat d'État ès Sciences, Université Paris-VI, 1983.
10. Y. SALHI, *Étude et réalisation de logiciels d'optimisation non contrainte avec dérivation numérique et estimation de la précision des résultats*, Thèse, Université Pierre-et-Marie-Curie, Paris-VI, 1985
11. R. S. STEPLEMAN et N. D. WINARSKY, *Adaptative Numerical Differentiation*, Mathematics of Computation, vol. 33, n° 148, octobre 1979, p. 1257-1264.
12. P. TOLLA, *Contribution à l'amélioration des logiciels de programmation mathématique en variables réelles*, Doctorat d'État ès Sciences, Université Paris-VI, 1983.
13. J. VIGNES, R. ALT et M. PICHAT, *Algorithmes numériques, analyse et mise en œuvre. Équations et systèmes non linéaires*, t. 2, Technip éd., 1980.
14. J. VIGNES, *Étude et mise en œuvre d'algorithmes de recherche d'un extremum d'une fonction de plusieurs variables*, Doctorat d'État ès Sciences, Faculté des Sciences de l'Université de Paris, 1969.
15. J. VIGNES et J. DUMONTET, *Détermination du pas optimal dans le calcul des dérivées sur ordinateur*, R.A.I.R.O. Anal. Num., vol. 11, n° 1, 1977, p. 13-25.
16. *Cray-1 Computer systems - Fortran (CFt)*, Reference Manual, Document Cray 2240009, Version E., 1980.