DEMONSTRATIO MATHEMATICA
Vol. XXXIX No 1 2006

N. Djeghaba, R. Benzine

ACCELERATION DE LA CONVERGENCE DE LA MÉTHODE DE LA PLUS FORTE PENTE

Abstract. Let (P) be the following problem of optimization without constraints

$$(P) \qquad \min\{f(x) : x \in \mathbb{R}^n\}.$$

We study in this paper an algorithm which acceleretes the convergence of the steepest descent method.

Résumé: Soit $f : \mathbb{R}^n \to \mathbb{R}$, et (P) le problème de minimisation sans contraintes suivant:

$$(P) \qquad \min\{f(x) : x \in \mathbb{R}^n\}.$$

On donne dans ce travail un algorithme de résolution du problème (P), qu'on a appelé L'Epsilon Steepest Descent dont le but est d'accelerer la convergence de la méthode de la plus forte pente.. On démontrera un résultat de convergence et on éffectuera ensuite des tests numériques.

1. Introduction

Soit $f : \mathbb{R}^n \to \mathbb{R}$, et (P) le problème de minimisation sans contraintes suivant:

$$(P) \qquad \min\{f(x) : x \in \mathbb{R}^n\}.$$

La méthode du gradient ou méthode de la plus forte pente, qui fut découverte par Cauchy en 1847, est parmi les anciennes méthodes utilisées pour résoudre le problème (P). L'origine ou la justification d'une telle appelation (méthode de la plus forte pente) vient du fait que si $x_k \in \mathbb{R}^n$ et si $\nabla f(x_k) \neq 0$, alors la direction $d_k = -\nabla f(x_k)$ est la meilleure direction de descente. En d'autres termes la décroissance de la fonction sera la plus forte en suivant la direction: $-\nabla f(x_k)$.

Cette méthode travaille de façon performante dans les premières étapes de l'algorithme. Malheuresement, dès qu'on s'approche du point station-

1991 *Mathematics Subject Classification*: 49D15, 65K05, 90C30.
Mots Clés: Steepest descent, acceleration de la convergence.

naire, elle devient trés lente. On peut expliquer intuitivement ce phénomène par les considérations suivantes

$$f(x_k + \lambda d) = f(x_k) + \lambda \nabla f(x_k)^t d + \lambda \|d\| \alpha(x_k; \lambda d), \tag{1}$$

avec $\alpha(x_k; \lambda d) \to 0$ quand $\lambda d \to 0$.

Si $d = -\nabla f(x_k)$, on obtient: $x_{k+1} = x_k - \lambda \nabla f(x_k)$ et (1) devient

$$f(x_{k+1}) - f(x_k) = \lambda \left[- \|\nabla f(x_k)\|^2 + \|\nabla f(x_k)\| \alpha(x_k; \lambda \nabla f(x_k)) \right]. \tag{2}$$

D'après l'expression (2), on voit que lorsque x_k s'approche d'un point stationnaire, et si f est continuement différentiable, alors $\|\nabla f(x_k)\|$ est proche de zéro. Donc le terme à droite s'approche de zéro, indépendemment de λ, et par conséquent $f(x_{k+1})$ ne s'éloigne pas beaucoup de $f(x_k)$ quand on passe du point x_k au point x_{k+1}.

Pour y remédier à ces inconvénients on essayera dans ce travail d'accélérer la convergence de la méthode de la plus forte pente. Pour cela on fera appel à un algorithme d'accélération de la convergence qui est l'epsilon Algorithme. On donnera un résultat de convergence et on comparera grace à des tests numériques notre algorithme avec la méthode de la plus forte pente et d'autres méthodes quasi-Newtonniennes (DFP et BFGS).

Signalons que d'autres méthodes d'acceleration de la méthode de la plus forte pente ont été proposées, notamment par Forsythe ([10]).

2. Accélération de la convergence en analyse numérique

Toutes les méthodes d'accélération de la convergence consistent à transformer une suite $\{s_n\}$ en une suite $\{v_n\}$ de même nature. Il faudra évidemment que $\{v_n\}$ converge vers la solution s de notre problème. Dans le cas d'une suite de nombres réels, on dit que $\{v_n\}$ converge plus vite que $\{s_n\}$ si

$$\lim_{n \to +\infty} \frac{v_n - s}{s_n - s} = 0.$$

On dira dans ce cas que l'on a accéléré la convergence de la suite $\{s_n\}$. La méthode qui transforme la suite $\{s_n\}$ en la suite $\{v_n\}$, sera une méthode d'accéléretion de la convergence.

Les possibilités d'accélération de la convergence d'une suite $\{s_n\}$ dépendront de la vitesse avec laquelle la suite $\{s_n\}$ converge. Si $\{s_n\}$ converge "vite", elle sera difficile à accélérer et cela ne présentera d'ailleurs pas beaucoup d'intérêt.

Voyons maintenant un algorithme assez puissant d'accélération de la convergence qui est l'ε−algorithme (voir [3] pour plus de détails). Il se

présente sous plusieurs formes selon la nature de la suite à accélérer: suite scalaire, suite vectorielle, suite matricielle.

2.1. L'ε-algorithme scalaire. L'ε-algorithme est dû à P. Wynn ([15]). Il repose sur de solides bases théoriques et ses applications sont extrêmement importantes. L'ε-algorithme est une généralisation d'une méthode célèbre d'accélération de la convergence: le procédé Δ^2 d'Aitken ([1]).

Dans cet algorithme, on calcule également des quantités avec deux indices $\varepsilon_k^{(n)}$. On utilise pour cela les relations

$$\varepsilon_{-1}^{(n)} = 0, \qquad \varepsilon_0^{(n)} = s_n \qquad n = 0, 1, \ldots \tag{3}$$

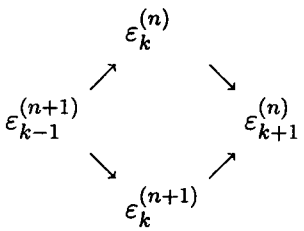
$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \frac{1}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}} \qquad n, k = 0, 1, \ldots$$

On place ces quantités dans un tableau à double entrée: le tableau ε. L'indice inférieur reste constant dans une colonne tandis que l'indice superieur reste constant dans une diagonale descente

$$\begin{array}{lllllll}
\varepsilon_{-1}^{(0)} = 0 & & & & & & \\
& \varepsilon_0^{(0)} = s_0 & & & & & \\
\varepsilon_{-1}^{(1)} = 0 & & \varepsilon_1^{(0)} & & & & \\
& \varepsilon_0^{(1)} = s_1 & & \varepsilon_2^{(0)} & & & \\
\varepsilon_{-1}^{(2)} = 0 & & \varepsilon_1^{(1)} & & \varepsilon_3^{(0)} & & \\
& \varepsilon_0^{(2)} = s_2 & & \varepsilon_2^{(1)} & & \varepsilon_4^{(0)} & \\
\varepsilon_{-1}^{(3)} = 0 & & \varepsilon_1^{(2)} & & \varepsilon_3^{(1)} & & . \\
& \varepsilon_0^{(3)} = s_3 & & \varepsilon_2^{(2)} & & \varepsilon_4^{(1)} & . \\
\varepsilon_{-1}^{(4)} = 0 & & \varepsilon_1^{(3)} & & \varepsilon_3^{(2)} & & . \\
. & \varepsilon_0^{(4)} = s_4 & . & \varepsilon_2^{(3)} & . & \varepsilon_4^{(2)} & . \\
. & . & . & . & . & . & . \\
. & . & . & . & . & . & . \quad .
\end{array}$$

La relation (3) relie des quantités situées aux quatre sommets d'un losange:

(4)



Si l'on connaît $\varepsilon_{k-1}^{(n+1)}$, $\varepsilon_k^{(n)}$ et $\varepsilon_k^{(n+1)}$, la relation (3) permet de calculer

$\varepsilon_{k+1}^{(n)}$; c'est ce que signifient les flèches du tableau (4). On dit que l'ε-algorithme est un algorithme de losange.

La relation (3) permet donc de progresser de la gauche vers la droite et de haut en bas dans le tableau ε, à partir des conditions initiales $\varepsilon_{-1}^{(n)} = 0$ et $\varepsilon_0^{(n)} = s_n$ pour $n = 0, 1, \ldots;$. Si l'on connaît s_0 et s_1 on peut calculer $\varepsilon_1^{(0)}$; si l'on connaît s_1 et s_2 on peut calculer $\varepsilon_1^{(1)}$; à partir de $\varepsilon_1^{(0)}$ et de $\varepsilon_1^{(1)}$ on obtiendra ensuite $\varepsilon_2^{(0)}$ et ainsi de suite.

2.2. L'ε-algorithme vectoriel. La forme vectorielle de l'ε-algorithme a également été étudiée par Wynn. Soit $\{s_n\}$ une suite de vecteurs de $\mathbb{C}^p$. Les règles de l'ε-algorithme vectoriel sont les suivantes

$$\varepsilon_{-1}^{(n)} = 0 \in \mathbb{C}^p \qquad \varepsilon_0^{(n)} = s_n \in \mathbb{C}^p \qquad n = 0, 1, \ldots \tag{5}$$

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \left[\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}\right]^{-1} \qquad n, k = 0, 1, \ldots$$

On voit que, pour pouvoir appliquer cet algorithme, il faut définir ce qu'on appelle l'inverse d'un vecteur. Etand donné $y \in \mathbb{C}^p$, on définit son inverse $y^{-1} \in \mathbb{C}^p$ par la relation

$$y^{-1} = \frac{\bar{y}}{(y, y)}, \tag{6}$$

$\bar{y}$ est le vecteur dont les composantes sont les nombres complexes conjugués des composantes du vecteur y, (y, y) désigne le produit scalaire dans $\mathbb{C}^p$, du vecteur par lui même. Si l'on désigne par $y_i (i = 1, \ldots, p)$, les composantes du vecteur y, alors

$$(y, y) = \sum_{i=1}^{p} y_i \bar{y}_i = \sum_{i=1}^{p} |y_i|^2 .$$

3. L'epsilon steepest descent algorithme

Pour construire notre Algorithme, on utilise l'$\varepsilon-$Algorithme scalaire d'ordre deux, c'est à dire $\varepsilon_2^{(n)}$. On pourra aussi utiliser l'ε-Algorithme vectoriel.

De façon plus précise, étant donnée une suite $\{s_n\}$, on montre dans ([3]), que les quantités $\varepsilon_2^{(n)}$ peuvent être calculées de la façon suivante

$$\varepsilon_2^{(n)} = \frac{s_n s_{n+2} - (s_{n+1})^2}{s_{n+2} - 2s_{n+1} + s_n}, \qquad n = 0, 1, 2, \ldots$$

On considère donc ici la deuxième colonne paire du tableau de l'ε-Algorithme scalaire, associé à la suite $\{s_n\}$.

REMARQUE. Pour calculer la quantité $\varepsilon_2^{(n)}$, il suffit d'avoir les élements: s_n, s_{n+1} et s_{n+2} de la suite $\{s_n\}$.

L'Algorithme ε-steepest descent

Initialisation: L'Algorithme commence par un point initial $x_0 \in \mathbb{R}^n$, les composantes de x_0 seront notées comme suit

$$x_0 = (x_0^1, x_0^2, \ldots, x_0^i, \ldots, x_0^n),$$

poser $k = 0$ et aller à étape principale.

Etape principale: Supposons qu'à l'étape k on ait le point x_k,

$$x_k = (x_k^1, x_k^2, \ldots, x_k^i, \ldots, x_k^n),$$

si $\|\nabla f(x_k)\| = 0$ stop; sinon poser

$$r_k = x_k,$$
$$r_k = (r_k^1, r_k^2, \ldots, r_k^i, \ldots, r_k^n).$$

Calculer par la méthode steepest descent les successeurs s_k et t_k de r_k, c'est à dire

$$s_k = (s_k^1, s_k^2, \ldots, s_k^i, \ldots, s_k^n),$$
$$t_k = (t_k^1, t_k^2, \ldots, t_k^i, \ldots, t_k^n),$$
$$s_k = r_k - \lambda_k \nabla f(r_k),$$

λ_k solution optimale de la recherche linéaire exacte

$$\underset{\lambda \geq 0}{\text{Minimiser}} f\left(r_k - \lambda \nabla f(r_k)\right)$$
$$t_k = s_k - \beta_k \nabla f(s_k),$$

β_k solution optimale de la recherche linéaire exacte

$$\underset{\beta \geq 0}{\text{Minimiser}} f\left(s_k - \beta \nabla f(s_k)\right).$$

Calculer

$$t_k^i - 2s_k^i + r_k^i;\ i = 1, \ldots n.$$

Si

$$t_k^i - 2s_k^i + r_k^i \neq 0;\ i = 1, \ldots n$$

calculer

$$\varepsilon_2^k = \left(\varepsilon_2^{k,1}, \varepsilon_2^{k,2}, \ldots \varepsilon_2^{k,i}, \ldots \varepsilon_2^{k,n}\right),$$

avec

$$\varepsilon_2^{k,i} = \frac{r_k^i t_k^i - \left(s_k^i\right)^2}{t_k^i - 2s_k^i + r_k^i}, \qquad i = 1, 2, \ldots n.$$

Si $f(\varepsilon_2^k) < f(t_k)$ poser

$$x_k = \varepsilon_2^k.$$

Remplacer k par $k+1$ et aller à l'étape principale. Si $f(\varepsilon_2^k) \geq f(t_k)$ ou si $t_k^{i_0} - 2s_k^{i_0} + r_k^{i_0} = 0$; $i_0 \in \{1, \ldots n\}$, poser

$$x_k = t_k.$$

Remplacer k par $k+1$ et aller à l'étape principale.

4. Convergence de l'epsilon steepest descent algorithme: cas des recherches linéaires exactes

Maintenant nous sommes en mesure de donner et démontrer le théorème de convergence suivant:

THÉORÈME 1. *Soit $f : \mathbb{R}^n \to \mathbb{R}$ telle que $f \in C^1(\mathbb{R}^n)$ et (P) le problème d'optimisation sans contraintes suivant:*

$$(P) \qquad \min\{f(x) : x \in \mathbb{R}^n\}.$$

On suppose que l'ensemble $\delta(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0),\ x_0 \in \mathbb{R}^n\}$ est borné. Soit $\{x_n\}_{n\in\mathbb{N}}$ la suite génerée par l'algorithme epsilon steepest descent. Soit x^* une limite d'une sous suite convrgente de $\{x_n\}_{n\in\mathbb{N}}$. Alors $\nabla f(x^*) = 0$.

Preuve. Supposons que l'algorithme génère une suite infinie $\{x_n\}_{n\in\mathbb{N}}$, car dans le cas contraire, il s'arrêterait dans un nombre fini d'itérations en un poit x tel que $\nabla f(x) = 0$, et le problème serait résolu. Soit $\{x_n\}_{n\in\mathbb{N}_1}$, $\mathbb{N}_1 \subset \mathbb{N}$, une sous suite convergente de $\{x_n\}_{n\in\mathbb{N}}$, de limite x^*. Montrons que $x^* \in \Omega$, avec $\Omega = \{x \in \mathbb{R}^n : \nabla f(x) = 0\}$.

Supposons le contraire, c'est à dire que $x^* \notin \Omega$. La suite $\{x_n\}_{n\in\mathbb{N}}$ pourrait être définie comme suit, x_0 étant un point initial, si x_n est connu à l'itération n, x_{n+1} sera défini comme suit:

$$x_{n+1} \in A(x_n).$$

A étant une fonction multivoque définie de la façon suivante:

$$A = C \circ B = C \circ B_2 \circ B_1$$

avec

$$\begin{aligned} B_1 &: \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n \\ x &\to B_1(x) = (x, x - \lambda_x \nabla f(x)); \\ \lambda_x \text{ vérifiant} &: f(x - \lambda_x \nabla f(x)) \leq f(x - \lambda \nabla f(x)) : \lambda \geq 0, \end{aligned}$$

$$\begin{aligned} B_2 &: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \\ (x, y) &\to B_2(x, y) = (x, y, y - \lambda_y \nabla f(y)); \\ \lambda_y \text{ vérifiant} &: f(y - \lambda_y \nabla f(y)) \leq f(y - \lambda \nabla f(y)) : \lambda \geq 0 \end{aligned}$$

et

$$C : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$$
$$(x, y, z) \to C(x, y, z) = w = (w^1, \ldots, w^n),$$

avec

$$w^i = \begin{cases} \frac{x^i z^i - (y^i)^2}{z^i - 2y^i + x^i}, \ si\ z^i - 2y^i + x^i \neq 0, \\ z^i, \ \text{sinon}, \end{cases}$$
$$x = (x^1, \ldots, x^n), \quad y = (y^1, \ldots, y^n), \quad z = (z^1, \ldots, z^n).$$

Il n'est pas difficile de montrer que l'application multivoque B est fermée en tout point $x \notin \Omega$ (voir [2] pour plus de détails). D'autre part f étant continue, alors

$$\lim_{n \to \infty,\, n \in \mathbb{N}_1} f(x_n) = f(x^*).$$

Montrons que

$$\lim_{n \to \infty} f(x_n) = f(x^*).$$

En effet, $\varepsilon > 0$ donné, il existe $n_0 \in \mathbb{N}_1$ tel que

$$|f(x_n) - f(x^*)| = f(x_n) - f(x^*) < \varepsilon, \ \text{pour } n \geq n_0, \ n \in \mathbb{N}_1.$$

En particulier pour $n = n_0$

$$f(x_{n_0}) - f(x^*) < \varepsilon.$$

Soit maintenant $n > n_0$, $n \in \mathbb{N}$. Il est facile de voir que par construction, la suite $\{f(x_n)\}_{n \in \mathbb{N}}$ est strictement décroissante. Donc

$$f(x_n) < f(x_{n_0}),$$

et par conséquent

$$f(x_n) - f(x^*) = f(x_n) - f(x_{n_0}) + f(x_{n_0}) - f(x^*) < 0 + \varepsilon = \varepsilon.$$

Ceci implique que

$$\lim_{n \to \infty} f(x_n) = f(x^*). \tag{7}$$

Considérons maintenant la suite $\{x_{n+1}\}_{n \in \mathbb{N}_1}$. D'après l'algorithme et les définitions données aux fonctions mutivoques A, C, B, B_1 et B_2 nous avons

$$x_{n+1} = C(x_n, y_n, z_n), \quad (x_n, y_n, z_n) \in B(x_n), \tag{8}$$

(dans l'algorithme nous avons $x_n = r_n$, $y_n = s_n$, $z_n = t_n$). Remarquons que les suites $\{x_n\}, \{y_n\}, \{z_n\}$ et $\{x_{n+1}\}$ appartiennent à l'ensemble compact

$\delta(x_0)$. Il existe alors $\mathbb{N}_2 \subset \mathbb{N}_1$ tels que

$$x_n \underset{n\to\infty,\, n\in\mathbb{N}_2}{\longrightarrow} x^* \tag{9}$$

$$y_n \underset{n\to\infty,\, n\in\mathbb{N}_2}{\longrightarrow} y^* \tag{10}$$

$$z_n \underset{n\to\infty,\, n\in\mathbb{N}_2}{\longrightarrow} z^* \tag{11}$$

$$x_{n+1} \underset{n\to\infty,\, n\in\mathbb{N}_2}{\longrightarrow} \widehat{x}. \tag{12}$$

Rappelons que B est fermée au point x^* (que nous avions supposé ne pas appartenir à Ω). Les relations (9), (10), (11), (12) et la définition de la fermeture de B au point x^* donnent

$$(x^*, y^*, z^*) \in B(x^*). \tag{13}$$

Puisque $x^* \notin \Omega$ ($\nabla f(x^*) \neq 0$), alors la direction $-\nabla f(x^*)$ est une direction de descente. Les définitions des fonctions multivoques B, B_1 et B_2 impliquent que

$$f(y^*) < f(x^*) \tag{14}$$

$$f(z^*) < f(x^*). \tag{15}$$

Maintenant, étant donné que

$$x_{n+1} = C(x_n, y_n, z_n), \tag{16}$$

alors la construction même de l'algorithme donne

$$f(x_{n+1}) \leq f(x_n) \tag{17}$$

$$f(x_{n+1}) \leq f(y_n) \tag{18}$$

$$f(x_{n+1}) \leq f(z_n). \tag{19}$$

Soit en passant à la limite quand $n \to \infty$, $n \in \mathbb{N}_2$

$$\begin{aligned} f(\widehat{x}) &\leq f(x^*) \\ f(\widehat{x}) &\leq f(y^*) \\ f(\widehat{x}) &\leq f(z^*). \end{aligned} \tag{20}$$

Les relations (14) et (15) impliquent

$$f(\widehat{x}) < f(x^*). \tag{21}$$

Puisque

$$f(x_{n+1}) \underset{n\to\infty,\, n\in\mathbb{N}_2}{\longrightarrow} f(\widehat{x}) \tag{22}$$

et

$$f(x_n) \underset{n\to\infty,\, n\in\mathbb{N}}{\longrightarrow} f(x^*) \quad \text{(voir (7))},$$

alors on aurait du avoir

$$f(\widehat{x}) = f(x^*). \tag{23}$$

La relation (23) est en contradiction avec la relation (21). Donc $x^* \in \Omega$, ou encore $\nabla f(x^*) = 0$. ∎

5. Résultats numériques

Nous avons effectué des tests numériques sur la fonction de Rosenbrock. Cette fonction est connue en littérature comme étant trés instable, et face à la méthode de la plus forte pente, elle présente un phénomène de Zigzaguing assez poussé. Les tests numériques que nous présentons ici sont appliqués à la même fonction, en utilisant la méthode $\varepsilon-$steepest descent (voir tableaux 1, 2, 3). Ils sont tres significatifs en ce qui concerne l'acceleration de la convergence. D'autres tests sur des fonctions assez classiques sont présentés. Nous avons essentiellement utilisé des recherches linéaires inexactes. Enfin nous avons comparé l'$\varepsilon-$steepest descent algorithme avec l'algorithme de la steepest descent et d'autres algorithmes quasi-Newotoniens (DFP et BFGS) considérés par quelques auteurs comme étant des accélerations de la méthode de la plus forte pente.

5.1. Fonctions tests. Les résultats numériques ont été obtenus en implémentant l'algorithme de steepest et epsilon steepest descent sur les fonctions tests suivantes :

PROBLÈME 1: (la fonction Rosenbrock)

$$f(x) = \sum_{i=1}^{n-1} \left\{100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right\},$$

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

PROBLÈME 2: (la fonction de Oren)

$$f(x) = \Big[\sum_{i=1}^{n} i x_i^2\Big]^2, \; x_0 = (1, 1, \ldots, 1), \; x^* = (0, 0, \ldots, 0), \; f(x^*) = 0.$$

PROBLÈME 3: (la fonction Pen1)

$$f(x) = \sum_{i=1}^{n}(x_i - 1)^2 + 10^{-3}\Big(\sum_{i=1}^{n} x_i^2 - 0,25\Big)^2$$

$x_0 = (\frac{1}{n+1}, \ldots, \frac{n}{n+1})$, $x^* = (1, -1, 1, -1, \ldots)^t$ pour $n = 50, 100$.

PROBLÈME 4: (la fonction trigonométrique)

$$f(x) = \sum_{i=1}^{n} \Big[n + i(1 - \cos x_i) - \sin x_i - \sum_{j=1}^{n} \cos x_j\Big]^2$$

$x_{0=[1/5n,\ldots,1/5n]}$, $x^* = [0, 0, 0, \ldots, 0]$, $f(x^*) = 0$.

5.2. Résultats numériques. Dans cette partie, on va adopter les notations suivantes:

n: la taille du problème

N: nombre des itérations

F_* : la valeur de la fonction objective

$\|\nabla F\|$: la norme du gradiant

5.2.1. *Résultats numériques pour la fonction de Rosenbrock*

Tableau 1: ε-steepest descent avec la recherche linéaire d'Armijo

n	N	$\| \nabla F \|$	F_*
2	646	$1,050 \times 10^{-5}$	$8,7210 \times 10^{-11}$
10	696	$1,0485 \times 10^{-5}$	$8,6373 \times 10^{-11}$
100	768	$1,0336 \times 10^{-5}$	$8,3944 \times 10^{-11}$
1000	838	$1,0492 \times 10^{-5}$	87060×10^{-11}
2000	860	$1,0347 \times 10^{-5}$	$8,5433 \times 10^{-11}$
10000	910	$1,0343 \times 10^{-5}$	$8,4610 \times 10^{-11}$

Tableau 2: $\varepsilon-$ steepest descent avec la recherche linéaire de Goldestein

n	N	$\| \nabla F \|$	F_*
2	88	$1,050 \times 10^{-5}$	$1,5602 \times 10^{-10}$
10	98	$1,8424 \times 10^{-5}$	$1,2003 \times 10^{-11}$
100	100	$1,8792 \times 10^{-5}$	$2,00508 \times 10^{-11}$
1000	108	$1,4885 \times 10^{-5}$	$1,4885 \times 10^{-10}$
2000	114	$9,9578 \times 10^{-5}$	$9,5439 \times 10^{-11}$
10000	122	$7,2350 \times 10^{-5}$	$3,03915 \times 10^{-11}$

Tableau 3: $\varepsilon-$ steepest descent avec la recherche linéaire de Wolfe

n	N	$\| \nabla F \|$	F_*
2	117	$2,8950 \times 10^{-5}$	$4,7003 \times 10^{-10}$
10	122	$1,4885 \times 10^{-5}$	$1,9575 \times 10^{-10}$
100	157	$1,0709 \times 10^{-5}$	$1,0424 \times 10^{-10}$
1000	202	$5,9882 \times 10^{-5}$	$1,4082 \times 10^{-10}$
2000	217	$1,1304 \times 10^{-5}$	$1,13846 \times 10^{-10}$
10000	151	$1,1027 \times 10^{-5}$	$3,03915 \times 10^{-10}$

5.2.2. *Résultats numériques pour la fonction de Oren*

Tableau 4: ε−steepest descent avec la recherche linéaire d'Armijo

n	N	$\| \nabla F \|$	F_*
2	6	$0{,}90\times10^{-7}$	$0{,}61\times10^{-10}$
10	12	$0{,}68\times10^{-6}$	$0{,}41\times10^{-9}$
100	46	$0{,}88\times10^{-5}$	$0{,}15\times10^{-7}$
1000	96	$0{,}38\times10^{-5}$	$0{,}68\times10^{-8}$
2000	136	$0{,}43\times10^{-5}$	$0{,}99\times10^{-8}$
10000	440	$0{,}68\times10^{-5}$	$0{,}12\times10^{-8}$

Tableau 5: steepest descent avec la recherche linéaire d'Armijo

n	N	$\| \nabla F \|$	F_*
2	6	$0{,}90\times10^{-7}$	$0{,}61\times10^{-10}$
10	14	$0{,}14\times10^{-5}$	$0{,}21\times10^{-8}$
100	67	$0{,}69\times10^{-5}$	$0{,}16\times10^{-7}$
1000	477	$0{,}41\times10^{-5}$	$0{,}113\times10^{-9}$
2000	1061	$0{,}18\times10^{-5}$	$0{,}22\times10^{-10}$
10000	N≥5000	/	/

5.2.3. *Résultats numériques pour la fonction Pen1*

Tableau 6: ε−steepest descent avec la recherche linéaire d'Armijo

n	N	$\| \nabla F \|$	F_*
50	6	$0{,}42\times10^{-6}$	$0{,}21\times10^{+1}$
100	7	$0{,}78\times10^{-5}$	$0{,}74\times10^{+1}$
1000	24	$0{,}81\times10^{-5}$	$0{,}29\times10^{+3}$
3000	22	$0{,}35\times10^{-5}$	$0{,}13\times10^{+4}$
10000	31	$0{,}97\times10^{-5}$	$0{,}57\times10^{+4}$

Tableau 7 : steepest descent avec la recherche linéaire d'Armijo

n	N	$\| \nabla F \|$	F_*
50	10	$0{,}69\times10^{-5}$	$0{,}21\times10$
100	18	$0{,}64\times10^{-5}$	$0{,}74\times10^{+1}$
1000	23	$0{,}54\times10^{-5}$	$0{,}29\times10^{+3}$
3000	N≥ 500	$0{,}15\times10^{-4}$	$0{,}13\times10^{+4}$
10000	N≥ 500	/	/

5.2.4. *Résultats numériques pour la fonction trigonométrique*

Tableau 8 : $\varepsilon-$steepest descent avec la recherche linéaire d'Armijo

n	N	$\| \nabla F \|$	F_*
10	5	$0{,}14\times10^{-6}$	$0{,}49\times10^{-14}$
50	4	$0{,}27\times10^{-5}$	$0{,}18\times10^{-11}$
100	4	$0{,}17\times10^{-5}$	$0{,}69\times10^{-12}$
1000	4	$0{,}47\times10^{-6}$	$0{,}54\times10^{-13}$

Tableau 9 : steepest descent avec la recherche linéaire d'Armijo

n	N	$\| \nabla F \|$	F_*
10	4	$0{,}12\times10^{-7}$	$0{,}38\times10^{-16}$
50	4	$0{,}2\times10^{-8}$	$0{,}99\times10^{-18}$
100	3	$0{,}87\times10^{-5}$	$0{,}19\times10^{-10}$
1000	3	$0{,}26\times10^{-5}$	$0{,}17\times10^{-11}$

5.2.5. *Comparaison avec la méthode DFP et BFGS.* Nous allons maintenant comparer l'epsilon steepest descent algorithme avec deux autres algorithmes quasi newtoniens tres celèbres qui sont l'algorithme DFP et BFGS (voir [4], [5], [6], [7], [8], [9], [11], [13], [14], [15] pour plus de détails). La comparaison se fera avec la fonction test de Oren.

Tableau 10: $\varepsilon-$steepest descent avec la recherche linéaire d'Armijo

n	N	$\| \nabla F \|$	F_*
2	6	$0{,}90\times10^{-7}$	$0{,}61\times10^{-10}$
10	12	$0{,}68\times10^{-6}$	$0{,}41\times10^{-9}$
100	46	$0{,}88\times10^{-5}$	$0{,}15\times10^{-7}$
1000	96	$0{,}38\times10^{-5}$	$0{,}68\times10^{-8}$
2000	136	$0{,}43\times10^{-5}$	$0{,}99\times10^{-8}$
10000	440	$0{,}68\times10^{-5}$	$0{,}12\times10^{-8}$

Tableau 11: DFP et BFGS avec recherche linéaire d'Armijo

	DEP		BFGS	
n	N	F_*	N	F_*
2	4	2.93×10^{-9}	4	2.70×10^{-9}
4	10	1.11×10^{-8}	10	7.81×10^{-9}
30	91	2.40×10^{-9}	68	9.55×10^{-9}
100	229	4.44×10^{-9}	213	5.88×10^{-9}

References

[1] A. C. Aitken, *On Bernoulli's numerical solution of algebraic equations*, Proc. Roy. Soc. Edinburgh, 46 (1926), 289–305.

[2] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programing*, John Wiley & Sons, New York, (1993).

[3] C. Brezinski, *Acceleration de la convergence en analyse numérique*, Lecture Notes in Mathematics, 584, Springer Verlag (1977).

[4] C. G. Broyden, J. E. Dennis, Jr., J. J. Moré, *On the local and superlinear convergence of quasi-Newton methods*, J. Inst. Math. Appl. 12 (1973), 223–246.

[5] J. E. Dennis, Jr., J. J. Moré, *A characterization of superlinear convergence and its application to quasi-Newton methods*, Math. Comp. 28 (1974), 549–560.

[6] J. E. Dennis, J. J. Moré, *Quasi-Newton methods, motivation and theory*, SIAM Rev. 19 (1977), 46–89.

[7] L. C. W. Dixon, *Variable metric algorithms : necessary and sufficient conditions for identical behavior on nonquadratic functions*, J. Opt. Theory Appl. 10 (1972), 34–40

[8] R. Fletcher, *Practical Methods of Optimization*, Second Edition, John Wiley & Sons, Chichester, 1987.

[9] R. Fletcher, *An Overview of Unconstrained Optimization*, in Algorithms for Continuous Optimization: the State of Art, E .Spedicato, ed., Kluwer Academic Publishers, 1994.

[10] G. E. Forsythe, *On the asymptotic directions of the s-dimentional optimum gradient method*, Num. Math. 11, pp. 57–76.

[11] P. E. Gill, W. Marray, *Quasi-Newton Methods for unconstrained optimization*, J. Inst. Math. Appl. 9 (1972), 91–108.

[12] A. Griewank, *The global convergence of partitioned BFGS on problems with convex decompositions and Lipschitz gradients*, Math. Prog. 50 (1991), 141–175.

[13] M. J. D. Powell, *On the convergence of the variable metric algorithms*, J. Inst. Math. Appl. 7 (1971), 21–36.

[14] M. J. D. Powell, *Some global convergence properties of variable metric algorithms for minimization without exact line searches*, in Nonlinear Programming, SIAM -AMS Proceedings, Vol. IX, R. W. Cottle, and C. E. Lemke (eds.), SIAM 1976.

[15] P. Wynn, *On a device for computing the $e_m(S_n)$ transformation*, M.T.A.C., 10, (1956), 91–96.

DÉPARTEMENT DE MATHÉMATIQUES
UNIVERSITÉ BADJI MOKHTAR ANNABA
B.P.12, 23000 ANNABA. ALGÉRIE
E-mail: rabenzine@yahoo.fr

Received August 13, 2004.