Table des matières

Introduction Générale

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

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

où $f : \mathbb{R}^n \to \mathbb{R}$ est continument différentiable.
L'optimisation non linéaire sans contraintes est utilisée dans des domaines variés, et l'étude des algorithmes et méthodes qui traitent ces problèmes est importante.
Pour résoudre le problème (P). La majorité des méthodes génèrent une suite $\{x_k\}_{k\in\mathbb{N}}$ de cette façon :

$$x_{k+1} = x_k + \alpha_k d_k$$

où d_k est une direction de descente et α_k est le pas obtenu en effectuant une optimisation unidimensionnelle .
Notons que

$$g_k = \nabla f(x_k)$$

dans la méthode du gradient les directions de descente sont de la forme :

$$d_k = -g_k.$$

Les propriétés de convergence des méthodes à directions de descente et recherches linéaires dépendent du bon choix de d_k et du pas α_k . L'angle que fait la direction d_k et la direction du gradient $-g_k$ est fondamental . C'est pour cela qu'on définit :

$$\cos(\theta_k) = \frac{-d_k^T g_k}{\|g_k\| \, \|d_k\|}.$$

On choisit α_k de sorte que la fonction f décroisse, mais en même il faut que ce calcul ne soit pas coûteux en temps et en mémoire . Le choix optimal est obtenu en choisissant α comme solution optimale de la fonction d'une variable $\varphi(\alpha)$ définie par :

$$\varphi(\alpha) = f(x_k + \alpha d_k).$$

Les recherches linéaires exactes consistent à calculer α_k comme solution du problème unidimensionnel suivant :

$$f(x_k + \alpha_k d_k) = \min\{f(x_k + \alpha d_k) \quad : \alpha > 0\}$$

mais ces recherches sont difficiles à réaliser en pratique, aussi ils sont coûteuses en temps et en mémoire .

C'est pour cela nous allons améliorer ce travaille en choisissant α_k vérifiant les deux conditions suivantes :

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k g_k^t d_k \tag{1}$$

$$g(x_k + \alpha_k d_k)^t d_k \geq c_2 g_k^t d_k \tag{2}$$

où $0 < c_1 < c_2 < 1$. La première relation (0.1) (*condition d'Armijo* $[5]$), assure que la fonction décroit suffisamment. La seconde condition (0.2) prévient que le pas α_k devienne très petit . Les deux conditions (0.1) et (0.2) s'appellent conditions de Wolfe.
Pour résoudre les problèmes du type (P) , on peut citer la méthode du gradient ou méthode de la plus forte pente $[11]$. Cette méthode présente le grand avantage d'avoir le meilleure décroissance à partir d'un point x_k, et malgré cela elle être très lente au voisinage des points stationnaires . Ils existe beaucoup de méthodes qui y remédient à ce problème, au lieu de considérer $d_k = -\nabla f(x_k)$, on peut se déplacer le long de $d_k = -D_k \nabla f(x_k)$
($[8]$, $[9]$, $[10]$, $[13]$, $[15]$, $[16]$, $[17]$, $[19]$, $[23]$, $[24]$, $[25]$, $[26]$, $[27]$, $[29]$, $[31]$, $[32]$, $[33]$) , ou bien le long $d_k = -g_k + h_k$ ($[20]$, $[21]$, $[22]$, $[23]$) , où D_k est une matrice choisie convenablement et h_k est un vecteur approprié .
Dans $[18]$ et $[33]$, Benzine, Djeghaba et Rahali ont essayé de résoudre ce problème par une autre méthode, en accélérant la convergence de la méthode du gradient.

Pour arriver à ce but, ils ont élaboré un nouveau algorithme qu'ils ont nommé l'epsilon steepest descent algorithm, dans lequel la formule de Florent Cordelier et celle de Wynn $([12], [36], [37])$ jouent un rôle essentiel . Ils ont aussi prouvé la convergence globale en utilisant des recherches linéaires exactes et d'Armijo.
D'autre part Degaichia et Benzine ont élaboré un autre algorithme qui est Wolfe $\varepsilon-$steepest descent algorithm $[14]$.
Dans cette thèse on va traité le problème d'accélération de la convergence, alors on a devisé notre travaille en quatre chapitres. Dans le chapitre 1 on commence par donner les motivations qui ont permis l'élaboration des méthodes d'accélération de la convergence, on présente ensuite des exemples de ces méthodes. On a deux classes de méthodes d'accélération de la convergence : linéaire, comme les procédés de sommation linéaire, et non linéaire, comme le procédé d'extrapolation de Richardson, le procédé Δ^2 D'aitken, l'$\varepsilon-$algorithme. Pour qu'il y ait convergence des procédés de sommation linéaire, des conditions suffisantes sont données dans le théorème de Toepliz. On donnera des résultats de convergence pour le procédé Δ^2 D'aitken ainsi que pour l'$\varepsilon-$algorithme scalaire et l'$\varepsilon-$algorithme vectoriel normé . On donnera des exemples dus à Brezinski de suites, dont la convergence est accélérée par l'$\varepsilon-$algorithme scalaire et par ses deux généralisations.

Dans le chapitre 2 on donne les définitions et les propriétés générales des problèmes d'optimisation sans contraintes . On définira la notion de direction de descente, notion très importante qui nous servira par la suite à construire les algorithmes d'optimisation sans contraintes, et on donne en détaille les conditions nécessaires et suffisantes. la partie suivante est consacrée à l'étude des minimums de fonctions d'une variable réelle ou recherche linéaire, car la plus part des algorithme d'optimisation non linière procède comme suit. Etant donné un point x_k, on cherche une certaine direction $d_k \in \mathbb{R}^n$ et un pas $\alpha_k \in \mathbb{R}$, ceci nous permet de trouver le successeur x_{k+1} de x_k par cette formule :

$$x_{k+1} = x_k + \alpha_k d_k$$

et le processus est ainsi répété . Le pas α_k est une solution optimale d'un problème de minimisation unidimensionnelle.Donc pour les problèmes de minimisation sans contraintes, on résout à chaque itération des sous problèmes de minimisation de fonctions d'une variable réelle.On éxpose en détaille les algorithmes et les propriétés de deux grandes classes de recherche linéaire. Il s'agit des recherches linéaires exactes et recherches linéaires inexactes. On s'intéresse à la méthode de dichotomie et la méthode du nombre d'or dans la première catégorie, et aux recherches linéaires d'Armijo, de Goldstein et de Wolfe pour la seconde catégorie, on éxpose aussi le théorème de Zoutendijk .
Puis dans la dérnière section on éxpose la notion de fonction multivoque, un outil théorique moderne et très puissant qui sert à étudier les propriétés d'un ensemble d'algorithmes au lieu d'un seul.
Le chapitre 3 est consacré à la méthode du gradient ou steepest descent. On explique l'origine d'une telle appellation et on montre que son défaut principal est sa convergence très lente au voisinage des points stationnaires. Des tests numérique effectués sur la fonction test de Rosembrok viennent confirmer ces faits.
Le chapitre 4 contient des résultats principales . On traite le problème de l'accélération de la convergence de la méthode du gradient avec l'outil d'accélération de la convergence donné au chapitre 1 : l'$\varepsilon-$algorithme.
La première section contient l'$\varepsilon-$steepest descent algorithm avec la recherche linéaire exacte, et le premier résultat de convergence de ce travaille. On montre dans ce résultatce que l'$\varepsilon-$steepest descent algorithm avec la recherche linéaire exacte converge vers un point stationnaire ou la solution optimale globale si la fonction objectif est convexe. On a utilisé les propriétés des fonctions multivoques pour arriver à notre but.
Ensuite dans les deux autres sections on accélère la convergence de la méthode du gradient en utilisant l'$\varepsilon-$steepest descent avec la recherche linéaire inexacte d'Armijo, et la recherche linéaire inexacte de Wolfe. , Ces deux sections contient aussi des résultats de convergence.

Dans la dernière section on compare avec des tests numériques entre ces algorithmes.

Chapitre 1

METHODES D'ACCELERATION DE LA CONVERGENCE EN ANALYSE NUMERIQUE

Pour résoudre certains problèmes numériquement nous devons utiliser des suites qui convergent vers la solution, mais parfois la convergence soit lente.
Les méthodes d'accélération de la convergence nous aident d'augmenter la vitesse d'une suite qui converge lentement en le remplaçant par une autre suite qui converge plus rapidement vers la même limite.
Soit $(S_n) = \{S_0, S_1, ..., S_n, ...\}$ une suite convergente vers la limite S dans le cas où (S_n) est une suite réelle, on dit que (V_n) converge plus vite que (S_n) si :

$$\lim_{n\to\infty} \frac{V_n - S}{S_n - S} = 0$$

alors on a accéléré la convergence de la suit (S_n) .

1.1 Les procédés de sommation linéaire

Les procédés de sommation linéaire ont été utilisés pour attribuer une limite aux séries divergentes (Hardy, Divergent series oxford press), ces procédés peuvent aussi être appliqués aux séries lentement convergentes dans le but de les accélérer.
Le principe de ces procédés est basé sur l'application d'une matrice triangulaire $A = (a_{i,j})$ qui transforme la suite originale (S_n) en une suite (V_n) comme suit :

$$\begin{pmatrix} V_0 \\ V_1 \\ . \\ . \end{pmatrix} = A \begin{pmatrix} S_0 \\ S_1 \\ . \\ . \end{pmatrix} \text{ avec } A = \begin{pmatrix} a_{00} & 0 & 0 & 0 & 0 \\ a_{10} & a_{11} & 0 & 0 & 0 \\ a_{20} & a_{21} & a_{22} & 0 & 0 \\ . & . & . & . & 0 \\ . & . & . & . & . \end{pmatrix} \tag{1.1}$$

On note par A le procédé de sommation linéaire .

Soit S la limite de la suite (S_n) , A est dit régulière si la suite transformée (V_n) converge vers la même limite S .

Pour affirmer la régularité de A on a ce théorème :

Théorème de Toepliz

A est régulière si :

i) $\sum_{k=0}^{\infty} |a_{nk}| < M \quad \forall n$

ii) $\lim_{n\to\infty} a_{nk} = 0 \qquad \forall k$

iii) $\lim_{n\to\infty} \sum_{k=0}^{\infty} a_{nk} = 1$

(sont des conditions suffisantes sans être nécessaire) .

Les principaux procédés régulières étudiés :

Le procédé $(H, 1)$ **de Hölder**

La matrice $A = (H, 1)$ est donnée par :

$$a_{ij} = \left\{ \begin{array}{ll} \frac{1}{i+1} & \text{pour } j = 1, ..., i \\ 0 & \text{pour } \ j > i \end{array} \right\}$$

alors

$$A = (H, 1) = \begin{pmatrix} \frac{1}{1} & 0 & 0 & 0 & . \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & . \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & . \\ . & . & . & . & 0 \\ . & . & . & . & . \end{pmatrix}$$

d'où la suite transformée est la suivante :

$$V_0 = S_0, V_1 = \frac{S_0 + S_1}{2}, V_2 = \frac{S_0 + S_1 + S_2}{3}, ...$$

On peut aussi utiliser le procédé généralisé $A^k = (H,k)$ (k entier) pour améliorer l'accélération de la convergence, pour $k=2$ on a $(H,2) = (H,1)(H,1)$, on obtient alors la nouvelle suite :

$$W_0 = V_0, W_1 = \frac{V_0+V_1}{2}, W_2 = \frac{V_0+V_1+V_2}{3}, ...$$

Exemple 1.1 *Soient les deux suites divergentes* :

a) $S_n = \{1,0,1,0,1...\}$ *telle que* $S_n = \sum_0^n (-1)^k = \frac{1+(-1)^n}{2}$ *on trouve* $V_n = \left\{1, \frac{1}{2}, \frac{2}{3}, \frac{2}{4}, \frac{3}{5}, \frac{3}{6}...\right\} \to \frac{1}{2}$

b) $S_n = \{1,0,1,0,1...\}$ *telle que* $S_n = \sum_0^n (-1)^k (k+1) = \frac{1+(-1)^n(2n-3)}{4}$ *on trouve successivement* :
$V_n = \left\{1, 0, \frac{2}{3}, 0, \frac{3}{5}, 0, \frac{4}{7}, 0, ...\right\}, W_n = \left\{1, \frac{1}{2}, \frac{5}{9}, \frac{5}{12}, \frac{34}{75}, \frac{34}{90}, ...\right\} \to \frac{1}{4}$

Le procédé (C,k) de Cesaro d'ordre k
Ca matrice (C,k) est donnée par :

$$(C,k) = (H,1) * L^{k-1}$$

où L est une matrice triangulaire inférieure gauche, avec tous les éléments égaux à l'unité .
Le procédé d'Euler
Dans ce procédé la matrice A est donnée par :

$$a_{ij} = \left\{ \begin{array}{cc} \frac{q^{i-j}}{(q+1)^i} C_i^j & \text{pour } j = 1, ..., i \\ 0 & \text{pour } j > i \end{array} \right\} \text{ avec } \sum_{j=0}^{\infty} a_{ij} = 1$$

q est choisi arbitrairement (souvent on prend $q=1$)

Remarque 1.1 :

1- *Les trois procédés précédents sont régulières* .

2- *Ces procédés vérifient* : $\lim_{n\to\infty} \sum_{k=0}^{\infty} a_{nk} = 1 \ \ \forall n$, *c'est la troisième condition du théorème de Toepliz* .

1.2 Le procédé d'extrapolation de Richardson

Soit (S_n) une suite qui converge vers S et soit (x_n) une suite auxiliaire strictement convergente et tendant vers le zéro lorsque n tend vers l'infini, le procédé de Richardson fait intervenir (x_n) pour obtenir un ensemble de suite $\left(V_k^{(n)}\right)$:

$$\begin{aligned} V_{k+1}^{(n)} &= \frac{x_n V_k^{(n+1)} - x_{n+k+1} V_k^{(n)}}{x_n - x_{n+k+1}} \qquad n,k = 0,1,... \\ V_0^{(n)} &= S_n \end{aligned} \tag{1.2}$$

On place ces quantités dans un tableau à double entrée :

$$\begin{array}{lllllll} V_0^{(0)} = S_0 & & & & & & \\ & V_1^{(0)} & & & & & \\ V_0^{(1)} = S_1 & & V_2^{(0)} & & & & \\ & V_1^{(1)} & & V_3^{(0)} & & & \\ V_0^{(2)} = S_2 & & V_2^{(1)} & & V_4^{(0)} & & \\ & V_1^{(2)} & & V_3^{(1)} & & . & \\ V_0^{(3)} = S_3 & & V_2^{(2)} & & V_4^{(1)} & & . \\ & V_1^{(3)} & & V_3^{(2)} & & . & \\ V_0^{(4)} = S_4 & . & V_2^{(3)} & . & V_4^{(2)} & & . \\ . & . & . & . & . & . & \\ . & . & . & . & . & . & . \end{array} \tag{1.3}$$

On peut calculer $V_{k+1}^{(n)}$ si on connait $V_k^{(n)}$ et $V_k^{(n+1)}$ de cette manière :

$$\begin{array}{ccc} V_k^{(n)} & \searrow & \\ & & V_{k+1}^{(n)} \\ V_k^{(n+1)} & \nearrow & \end{array} \tag{1.4}$$

La convergence de ce procédé est obtenue sous des conditions très restrictives sur les suites $\left(V_k^{(n)}\right)$, $\left(V_k^{(n+1)}\right)$ et sur la suit auxiliaire (x_n) comme décrit dans les théorèmes suivantes :

Théorème 1.1 [2] *Soit (x_n) une suite strictement décroissante de nombres réels positifs et tendant vers zéro lorsque n tend vers l'infini .*
Une condition nécessaire et suffisante pour que :

$$\lim_{k\to\infty} V_k^{(n)} = \lim_{k\to\infty} S_k$$

pour tout $n = 0,1,...$ et pour toute suite convergente (S_k) et qu'il existe $a > 1$ tel que :

$$\frac{x_n}{x_{n+1}} \geqslant a > 1 \quad , \; n = 0,1,..$$

Cette condition est aussi une condition nécessaire et suffisante pour que :

$$\lim_{n\to\infty} V_k^{(n)} = \lim_{n\to\infty} S_n$$

pour tout $k = 0,1,...$ et pour toute suite convergente (S_n) .

Remarque 1.2 *La condition du théorème ci-dessus entraine la convergence des colonnes et des diagonales du tableau* (1.3).

Théorème 1.2 [2] *Supposons que la condition du théorème précédent soit satisfait, alors une condition nécessaire et suffisante pour que la suite* $(V_{k+1}^{(n)})$ (k *fixé*) *converge plus vite que la suite* $(V_k^{(n)})$(k *fixé*) *est que* :

$$\lim_{n\to\infty} \frac{V_k^{(n+1)} - S}{V_k^{(n)} - S} = \lim_{n\to\infty} \frac{x_{n+k+1}}{x_n}.$$

1.3 Le procédé $\triangle^2$ d'Aitken

Ce procédé consiste à transformer la suite (S_n) en une suite notée $(V(S_n))$ définie par :

$$(V(S_n)) = \frac{S_n S_{n+2} - (S_{n+1})^2}{S_{n+2} - 2S_{n+1} + S_n} \quad , n = 0, 1, ... \tag{1.5}$$

avec

$$S_{n+2} - 2S_{n+1} + S_n \neq 0 \qquad \forall n$$

Notons que :

$$\triangle S_n = S_{n+1} - S_n \qquad \forall n \geq 0$$

alors on peut écrire $(V(S_n))$ de cette façon :

$$(V(S_n)) = S_{n+1} - \frac{\triangle S_{n+1}}{\frac{\triangle S_{n+1}}{\triangle S_n} - 1}.$$

Un résultat de convergence pour le procédé d'Aitken est donné par ce théorème :

Théorème 1.3 [2] *Soit* (S_n) *une suite qui converge vers la limite* S, *si*

$$\lim_{n\to\infty} \frac{S_{n+1} - S}{S_n - S} = \frac{\triangle S_{n+1}}{\triangle S_n} = \alpha \neq 1$$

alors $(V(S_n))$ *converge vers* S *plus rapidement que* (S_n) .

1.4 L'ε-algorithme

L'ε-algorithme est un algorithme non linéaire d'accélération de la convergence de suites numériques, il été proposé par P.Wynn en 1956, c'est l'un des algorithmes les plus connus et une généralisation de l'algorithme $\triangle^2$ d'Aitken.

1.4.1 L'ε-algorithme scalaire

Soit (S_n) une suite numérique dont on cherche à trouver sa limite S, l'ε-algorithme scalaire consiste à calculer des quantités à deux indices $\varepsilon_k^{(n)}$ à l'aide des relations suivantes :

$$\begin{aligned} \varepsilon_{-1}^{(n)} &= 0 \qquad \varepsilon_0^{(n)} = S_n \qquad n = 0, 1, ... \\ \varepsilon_{k+1}^{(n)} &= \varepsilon_{k-1}^{(n+1)} + \frac{1}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}} \qquad n, k = 0, 1, ... \end{aligned} \tag{1.6}$$

Les résultats de l'ε-algorithme sont présentés sous forme d'un tableau avec l'indice inférieur reste constant dans une colonne et l'indice supérieur reste constant dans une diagonale descente de cette façon :

$$\begin{array}{llllllll} \varepsilon_{-1}^{(0)} = 0 & & & & & & & \\ & \varepsilon_0^{(0)} = S_0 & & & & & & \\ \varepsilon_{-1}^{(1)} = 0 & & \varepsilon_1^{(0)} & & & & & \\ & \varepsilon_0^{(1)} = S_1 & & \varepsilon_2^{(0)} & & & & \\ \varepsilon_{-1}^{(2)} = 0 & & \varepsilon_1^{(1)} & & \varepsilon_3^{(0)} & & & \\ & \varepsilon_0^{(2)} = S_2 & & \varepsilon_2^{(1)} & & \varepsilon_4^{(0)} & & \\ \varepsilon_{-1}^{(3)} = 0 & & \varepsilon_1^{(2)} & & \varepsilon_3^{(1)} & & . & \\ & \varepsilon_0^{(3)} = S_3 & & \varepsilon_2^{(2)} & & \varepsilon_4^{(1)} & . & . \\ \varepsilon_{-1}^{(4)} = 0 & & \varepsilon_1^{(3)} & & \varepsilon_3^{(2)} & & . & . & . \\ . & \varepsilon_0^{(4)} = S_4 & . & \varepsilon_2^{(3)} & . & \varepsilon_4^{(2)} & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \end{array} \tag{1.7}$$

La relation (1.7) relie des quantités situées au quatre sommets d'un losange

$$\begin{array}{ccc} & \varepsilon_k^{(n)} & \\ \nearrow & & \searrow \\ \varepsilon_{k-1}^{(n+1)} & & \varepsilon_{k+1}^{(n)} \\ \searrow & & \nearrow \\ & \varepsilon_k^{(n+1)} & \end{array} \tag{1.8}$$

Généralisations

Première généralisation : En introduisant une suite auxiliaire (x_n) dans (1.7),on obtient les relations suivantes :

$$\begin{aligned} \varepsilon_{-1}^{(n)} &= 0 \qquad \varepsilon_0^{(n)} = S_n \qquad n = 0, 1, ... \\ \varepsilon_{k+1}^{(n)} &= \varepsilon_{k-1}^{(n+1)} + \frac{\triangle x_n}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}} \qquad n, k = 0, 1, ... \end{aligned} \tag{1.9}$$

Deuxième généralisation : En introduisant aussi une suite auxiliaire (x_n) dans (1.7),on obtient les relations suivantes :

$$\begin{aligned} \varepsilon_{-1}^{(n)} &= 0 \qquad \varepsilon_0^{(n)} = S_n \qquad n = 0, 1, ... \\ \varepsilon_{k+1}^{(n)} &= \varepsilon_{k-1}^{(n+1)} + \frac{\triangle x_{n+k}}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}} \qquad n, k = 0, 1, ... \end{aligned} \tag{1.10}$$

avec

$$\triangle x_{n+k} = x_{n+k+1} - x_{n+k} \qquad \forall n.$$

Convergence de l'ε-algorithme scalaire

Définition 1.1 *On dit que la suite (S_n) est totalement monotone si* :

$$(-1)^k \triangle^k S_n \geq 0 \qquad \forall n, k = 0, 1, ...$$

où

$$\triangle^0 S_n = S_n$$

et pour tout n :

$$\triangle^{k+1} S_n = \triangle^k S_{n+1} - \triangle^k S_n \qquad n, k = 0, 1, ...$$

Théorème 1.4 [2] *Soit (S_n) une suite qui converge vers S s'il existe deux constante $a \neq 0$ et b telle que la suite $\{aS_n + b\}$ soit totalement monotone alors* :

$$\begin{aligned} \lim_{n \longrightarrow \infty} \varepsilon_{2k}^{(n)} &= S \qquad k = 0, 1, ... \textit{fixé} . \\ \lim_{k \to \infty} \varepsilon_{2k}^{(n)} &= S \qquad n = 0, 1, ... \textit{fixé} . \end{aligned}$$

Définition 1.2 *Si la suite $((-1)^n S_n)$ est totalement monotone alors on peut dire que la suite (Sn) est totalement oscillante .*

Théorème 1.5 [2] *Si on applique l'ε-algorithme scalaire à une suite (S_n) qui converge vers S et s'il existe deux constantes $a \neq 0$ et b telle que la suite $\{aS_n + b\}$ soit totalement oscillante alors* :

$$\begin{aligned} \lim_{n \to \infty} \varepsilon_{2k}^{(n)} &= S \qquad k = 0, 1, ... \textit{fixé} . \\ \lim_{k \to \infty} \varepsilon_{2k}^{(n)} &= S \qquad n = 0, 1, ... \textit{fixé} . \end{aligned}$$

Remarque 1.3 *Les théorèmes* (1.5) *et* (1.6) *n'accélèrent pas la convergence, mais montrent que l'*ε*-algorithme converge.*
De plus on remarque que :

$$\lim_{n\to\infty} \frac{\varepsilon_{2k}^{(n)} - S}{S_{n+2k} - S} = 0 \qquad k = 1, 2, ...$$
$$\lim_{k\to\infty} \frac{\varepsilon_{2k}^{(n)} - S}{S_{n+2k} - S} = 0 \qquad n = 1, 2, ...$$

1.4.2 L'ε-algorithme vectoriel et l'ε-algorithme vectoriel normé

L'ε-algorithme vectoriel est analogue dans sa forme à l'ε-algorithme scalaire .
Soit $\{Sn\}$ une suite de vecteurs de C^p

$$\begin{aligned} \varepsilon_{-1}^{(n)} &= 0 \in C^p \qquad \varepsilon_0^{(n)} = S_n \in C^p \qquad n = 0, 1, ... \\ \varepsilon_{k+1}^{(n)} &= \varepsilon_{k-1}^{(n+1)} + \left[\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}\right]^{-1} \qquad n, k = 0, 1, ... \end{aligned} \tag{1.11}$$

tel que l'inverse d'un vecteur $y \in C^p$ est donné par :

$$y^{-1} = \frac{\bar{y}}{(y, y)_{c_p}} \in c^p \tag{1.12}$$

où $\| y \|$ est la norme vectoriel de y .
Convergence de l'ε-algorithme vectoriel normé (**vectoriel**)
L'ε -algorithme vectoriel est un cas particulier de l'ε -algorithme vectoriel normé, donc les théorèmes de convergence suivantes sont utilisables pour les deux algorithmes.
Hypothèses 1

1) $\{S_n\}$ une suite de vecteurs de C^p.

2) (a_n) une suite réelle avec $(an) \neq 0\ \forall n$ et $(\triangle a_n)$ converge vers zéro, supposons aussi qu'il existe deux constantes α et β telle que :

$$\alpha < 1 < \beta \text{ et } a_n \notin [\alpha, \beta] \quad \forall n \ .$$

3) La suite $\{S_n\}$ vérifie :

$$S_{n+1} - S = a_n (S_n - S), \qquad n = 0, 1, ...$$

Théorème 1.6 [2] *Si on applique l'*ε*-algorithme vectoriel normé* (*vectoriel*) *à la suite* $\{S_n\}$ *qui vérifie les hypothèses 1 précédentes et qui converge vers* S *alors* :

$$\lim_{n\to\infty} \varepsilon_2^{(n)} = S.$$

Si de plus la suite (a_n) *admet une limite alors* :

$$\lim_{n\to\infty} \frac{\| \varepsilon_2^{(n)} - S \|}{\| S_{n+1} - S \|} = 0 \ .$$

Hypothèses 2

1) $\{S_n\}$ une suite de vecteurs de C^p.

2) (e_n) est une suite de vecteur de C^p qui converge vers zéro lorsque n tend vers l'infini .

3) $y \in C^p$ et $y \neq 0$.

4) a est un nombre réel tel que $0 < a < 1$.

5) La suite $\{S_n\}$ vérifie :

$$S_n - S = a^n (y + e_n) , \qquad n = 0, 1, ...$$

avec $S \in C^p$.

Théorème 1.7 [2] *Si on applique l'*ε*-algorithme vectoriel normé (vectoriel) à la suite* $\{S_n\}$ *qui vérifie les hypothèses* 2 *précédentes alors* :

$$\lim_{n\to\infty} \varepsilon_2^{(n)} = S$$

et :

$$\lim_{n\to\infty} \frac{\| \varepsilon_2^{(n)} - S \|}{\| S_{n+k} - S \|} = 0, \forall k \geq 0 \ \textit{fixé} \ .$$

les théorèmes 1.6 et 1.7 montrent que l'ε-algorithme vectoriel normé (vectoriel) converge vers la limite de la suite original et que cette convergence est accélérée.

1.4.3 Exemples de suites accélérées par l'ε-algorithme scalaire et ses deux généralisations

Exemple1 : Cet exemple est dû à Brezinski .
Soit (S_n) la suite définie par :

$$S_n = 1 + 3. \exp(-1.4x_n) \quad \text{avec } x_n = 1.1^{n-1}.$$

Cette suite converge lentement vers1 .
On a :
$S_0 = 1.73979....; S_1 = 1.64314....; .....; S_4 = 1.38631....$
En appliquant l'$\varepsilon-$algorithme scalaire, on obtient :
$\varepsilon_4^{(0)} = 1.73799....$

et en appliqunt la première généralisation de l'ε-algorithme scalaire on obtient :
$\varepsilon_4^{(0)} = 1.00272....$
La deuxième généralisation de l'ε-algorithme scalaire donne :
$\varepsilon_4^{(0)} = 1.00224....$
Exemple 2 : Cet exemple est également dû à Brezinski .
Soit (S_n) une suite définie par :

$$S_n = n \sin\left(\frac{1}{n}\right) \text{ avec } x_n = \log(n+1).$$

Cette suite converge vers 1.
On obtient : $S_{37} = 0.99987....$
et $\varepsilon_{36}^{(0)} = 0.999938....$en appliquant l'$\varepsilon$-algorithme scalaire .
La première généralisaton de l'ε-algorithme scalaire donne $\varepsilon_{36}^{(0)} = 1.0000000027$, la deuxème généralisation de l'ε-algorithme scalaire donne $\varepsilon_{36}^{(0)} = 0.9999999976...$

Chapitre 2

OPTIMISATION SANS CONTRAINTES

Le problème que l'on étudie ici est celui de la recherche du minimum d'une fonction réelle f de n variables $x_1, x_2, ..., x_n$.

Définition 2.1 *Soit* $f : \mathbb{R}^n \to \mathbb{R}$ *qui à tout* $x \in \mathbb{R}^n$, $x = (x_1, x_2, ..., x_n)^t$, *associe la valeur réelle*

$$f(x) = f(x_1, x_2, ..., x_n) \ .$$

On cherche à résoudre le problème (P) :

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

Il s'agit donc de déterminer un point $\hat{x}$ *de* $\mathbb{R}^n$ *tel que* :

1. $\hat{x} \in \mathbb{R}^n$ *s'appelle solution minimum global de* (P) *si et seulement si*

   $$f(\hat{x}) \leq f(x) : \forall x \in \mathbb{R}^n$$

   $f(\hat{x})$ *s' appelle valeur minimum global .*

2. $\hat{x} \in \mathbb{R}^n$ *s'appelle solution minimum local de* (P) *si et seulement si il existe un voisinage* $V_\varepsilon(\hat{x})$ *tel que*

   $$f(\hat{x}) \leq f(x) : \forall x \in V_\varepsilon(\hat{x})$$

   $f(\hat{x})$ *s' appelle valeur minimum local .*

3. $\hat{x} \in \mathbb{R}^n$ *s'appelle solution minimum local strict de* (P) *si et seulement si il existe un voisinage* $V_\varepsilon(\hat{x})$ *tel que*

   $$f(\hat{x}) < f(x) : \forall x \in V_\varepsilon(\hat{x}), \ x \neq \hat{x}$$

   $f(\hat{x})$ *s'appelle valeur minimum local strict.*

2.1 Direction de descente

Définition 2.2 *Soit $f : \mathbb{R}^n \to \mathbb{R}$, $\hat{x} \in \mathbb{R}^n$, $d \in \mathbb{R}^n$ est dite direction de descente au point $\hat{x}$ si et seulement si il existe un nombre strictement positive $(\delta > 0)$ tel que :*

$$f(\hat{x} + \lambda d) < f(\hat{x}) \qquad : \forall \lambda \in]0, \delta[.$$

Donnons une condition suffisante pour que d soit une direction de descente.

Théorème 2.1 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ différentiable au point $\hat{x} \in \mathbb{R}^n$ et $d \in \mathbb{R}^n$ une direction vérifiant la condition suivante :*

$$f'(\hat{x}, d) = \nabla f(\hat{x})^t.d < 0$$

alors d est une direction de descente au point $\hat{x}$.

Preuve. f est différentiable au point $\hat{x}$ alors f continue et $\nabla f(\hat{x})$ existe, donc

$$f(\hat{x} + \lambda d) = f(\hat{x}) + \lambda \nabla f(\hat{x})^t.d + \lambda \parallel d \parallel \alpha(\hat{x}, \lambda d)$$

alors

$$f(\hat{x} + \lambda d) - f(\hat{x}) = \lambda \nabla f(\hat{x})^t.d + \lambda \parallel d \parallel \alpha(\hat{x}, \lambda d)$$

$$\Rightarrow \frac{f(\hat{x} + \lambda d) - f(\hat{x})}{\lambda} = \nabla f(\hat{x})^t.d + \parallel d \parallel \alpha(\hat{x}, \lambda d)$$

$$\Rightarrow \lim_{\lambda \to 0} \frac{f(\hat{x} + \lambda d) - f(\hat{x})}{\lambda} = \lim_{\lambda \to 0} \left(\nabla f(\hat{x})^t.d + \parallel d \parallel \alpha(\hat{x}, \lambda d) \right)$$

avec

$$\alpha(\hat{x}, \lambda d) \underset{\lambda \to 0}{\to} 0$$

donc

$$f'(\hat{x}, d) = \lim_{\lambda \to 0} \frac{f(\hat{x} + \lambda d) - f(\hat{x})}{\lambda} = \nabla f(\hat{x})^t.d < 0$$

la limite étant strictement négative, alors il existe un voisinage de zéro $V(0) =]-\delta, +\delta[$ tel que

$$\frac{f(\hat{x} + \lambda d) - f(\hat{x})}{\lambda} < 0 \quad , \forall \lambda \in]-\delta, +\delta[\tag{2.1}$$

la relation (2.1) est particulièrement vraie pour tout $\lambda \in]0, +\delta[$.on obtient le résultat cherché en multipliant la relation (2.1) par $\lambda > 0$. ■

2.2 Schéma général des algorithmes

Soit d_k une direction de descente aux point x_k on peut considérer le point x_{k+1} le successeur de x_k comme suit :

$$x_{k+1} = x_k + \lambda_k d_k \text{ , } \lambda_k \in]0, +\delta[\text{ .}$$

Démarrage : $x_0 \in \mathbb{R}^n, d_0$:

$$\nabla f(x_0)^t . d_0 < 0$$

$$x_1 = x_0 + \lambda_0 d_0$$

λ_0 vérifie :

$$f(x_0 + \lambda_0 d_0) < f(x_0)$$

Itération k :x_k, d_k telle que $\nabla f(x_k)^t . d_k < 0$ et λ_k telle que :

$$f(x_k + \lambda_k d_k) < f(x_k)$$

alors

$$x_{k+1} = x_k + \lambda_k d_k$$

Le choix de d_k, et λ_k permet de construire une multitude d'algorithmes d'optimisation.

Exemple de choix de directions de descente :

- Si on choisit : $d_k = -\nabla f(x_k)$,avec $\nabla f(\hat{x}_k) \neq 0$,on obtient la méthode de gradient.

Bien sur $d_k = -\nabla f(x_k)$ est une direction de descente, en effet :

$$\nabla f(x_k)^t d_k = \nabla f(x_k)^t (-\nabla f(x_k)) = -\nabla f(x_k)^t . \nabla f(x_k) = - \parallel \nabla f(x_k)^t \parallel^2 < 0$$

- Aussi si on choisit : $d_k = -(H(x_k))^{-1} \nabla f(x_k)$ tel que :

$H(x_k)$ la matrice Hessienne.$(H(x_k) \in \mathcal{M}_{n \times n})$, $\nabla f(x_k)$ le vecteur de gradient.$(\nabla f(x_k) \in \mathcal{M}_{n \times 1})$, on obtient la méthode de Newthon.
si la matrice $H(x_k)$ est définie positive, alors

$$\nabla f(x_k)^t d_k = -\nabla f(x_k)^t (H(x_k))^{-1} \nabla f(x_k) < 0$$

Exemple de choix de pas λ_k :
On choisit λ_k vérifier

$$f(x_k + \lambda_k d_k) \leq f(x_k + \lambda d_k), \qquad \forall \lambda \in]0, \delta[$$

la recherche d'une variable réelle λ_k qui s'appelle la recherche linéaire.

2.3 Conditions d'optimalité

Condition nécessaire d'optimalité du premier ordre

Théorème 2.2 *soit $f : \mathbb{R}^n \to \mathbb{R}$ différentiable au point $\hat{x} \in \mathbb{R}^n$, si $\hat{x}$ est une solution minimaux local, alors $\nabla f(\hat{x}) = 0$.*

Preuve. *$\hat{x}$ est une solution minimaux local, alors*

$$f(\hat{x}) \leq f(x), \qquad \forall x \in V(\hat{x}) \tag{2.2}$$

supposons le contraire $\nabla f(\hat{x}) \neq 0$,alors $-\nabla f(\hat{x})$ est une direction de descente alors $\exists \delta > 0$ tel que $\forall \alpha \in]0, \delta[$:

$$f(\hat{x} + \alpha(-\nabla f(\hat{x}))) < f(\hat{x})$$

on pose $\hat{x} + \alpha(-\nabla f(\hat{x}) = \bar{x}$ alors

$$f(\bar{x}) < f(\hat{x})$$

donc $\exists \bar{x} \in V(\hat{x})$ tel que :

$$f(\bar{x}) < f(\hat{x}) \tag{2.3}$$

la contradiction entre (2.2) et (2.3).

$$\text{d'ou} \begin{cases} f \text{ différentiable} \\ \text{et} \\ f(\hat{x}) \leq f(x), \qquad \forall x \in V(\hat{x}) \end{cases} \qquad \text{alors} \qquad \nabla f(\hat{x}) = 0. \ \blacksquare$$

Condition nécessaire d'optimalité du second ordre

Théorème 2.3 *soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois différentiable au point $\hat{x} \in \mathbb{R}^n$,si $\hat{x}$ est un minimaux local de (P) alors $\nabla f(\hat{x}) = 0$ et la matrice hessienne de f au point $\hat{x}$, qu'on note $H(\hat{x})$,est semi définie positive.*

Preuve. *soit $x \in \mathbb{R}^n$ quelconque, f étant deux fois différentiable au point $\hat{x}$ on aura pour tout $\lambda \neq 0$*

$$f(\hat{x} + \lambda x) = f(\hat{x}) + \frac{1}{2}\lambda^2 x^t H(\hat{x}) x + \lambda^2 \left\|x^2\right\| \alpha(\hat{x}, \lambda x), \qquad \alpha(\hat{x}, \lambda x) \underset{\lambda \to 0}{\to} 0$$

Ceci implique

$$\frac{f(\hat{x} + \lambda x) - f(\hat{x})}{\lambda^2} = \frac{1}{2} x^t H(\hat{x}) x + \left\|x^2\right\| \alpha(\hat{x}, \lambda x) \tag{2.4}$$

$\hat{x}$ est un optimum local, il existe alors $\delta > 0$ tel que

$$\frac{f(\hat{x} + \lambda x) - f(\hat{x})}{\lambda^2} \geq 0, \qquad \forall \lambda \in]-\delta, +\delta[$$

si on prend en considération (2.4) *et on passe à la limite quand* $\lambda \to 0, \lambda \neq 0$, *on obtient*

$$x^t H(\hat{x}) x \geq 0, \qquad \forall x \in \mathbb{R}^n.$$

■

Condition suffisante d'optimalité

Théorème 2.4 *soit* $f : \mathbb{R}^n \to \mathbb{R}$ *deux fois différentiable au point* $\hat{x} \in \mathbb{R}^n$, *Si* $\nabla f(\hat{x}) = 0$ *et* $H(\hat{x})$ *est définie positive alors* $\hat{x}$ *est un minimum local strict de* (P).

Preuve. f *étant deux fois différentiable au point* $\hat{x}$, *on aura pour tout* $x \in \mathbb{R}^n$

$$f(x) = f(\hat{x}) + \frac{1}{2}(x-\hat{x})^t H(\hat{x})(x-\hat{x}) + \|(x-\hat{x})\|^2 \alpha(\hat{x},(x-\hat{x})), \alpha(\hat{x},(x-\hat{x})) \underset{x\to\hat{x}}{\to} 0, (\nabla f(\hat{x}) = 0) \tag{2.5}$$

supposons que $\hat{x}$ *n'est pas un optimum local strict .*
Alors il existe une suite $\{x_k\}_{k\in N^*}$ *telle que* $x_k \neq \hat{x} : \forall k$ *et*

$$x_k \neq \hat{x} : \forall k, x_k \underset{k\to\infty}{\to} \hat{x} \text{ et } f(x_k) \leq f(\hat{x}). \tag{2.6}$$

Dans (2.5) *prenons* $x = x_k$, *division le tout par* $\|(x-\hat{x})\|^2$ *et notons* $d_k = \frac{(x_k-\hat{x})}{\|(x_k-\hat{x})\|}$, *on obtient*

$$\frac{f(x_k)-f(\hat{x})}{\|(x_k-\hat{x})\|^2} = \frac{1}{2} d_k^t H(\hat{x}) d_k + \alpha(\hat{x},(x_k-\hat{x})), \alpha(\hat{x},(x_k-\hat{x})) \underset{k\to\infty}{\to} 0. \tag{2.7}$$

(2.6) *et* (2.7) *impliquent*

$$\frac{1}{2} d_k^t H(\hat{x}) d_k + \alpha(\hat{x},(x_k-\hat{x})) \leq 0, \qquad \forall k.$$

d'autre part la suite $\{d_k\}_{k\in N^*}$ *est bornée* $(\| d_k \| = 1, \forall n)$. *Donc il existe une sous suite* $\{d_k\}_{k\in N_1 \subset N}$ *telle que*

$$d_k \underset{k\to\infty, k\in N_1}{\to} \bar{d}.$$

Finalement lorsque $k \to \infty, k \in N_1$, *on obtient*

$$\frac{1}{2} \bar{d}^t H(\hat{x}) \bar{d} \leq 0.$$

La dernière relation et le fait que $\bar{d} \neq 0$ $(\| \bar{d} \| = 1)$ *impliquent que la matrice hessienne* $H(\hat{x})$ *n'est pas définie positive. Ceci est en contradiction avec l'hypothèse.* ■

2.4 Optimisation unidimensionnelle

L'optimisation unidimensionnelle (recherche linéaire) consiste à trouver λ_k de façon à diminuer la fonction f Suffisamment le long de cette direction.
Ce " suffisamment " sera quantifié dans la suite dans la description des conditions dites d'Armijo, Wolfe, Goldstein & Price (recherches linéaires inexactes).
Mais d'abord on expose le principe de méthode de descente :

2.4.1 Principe de méthode de descente

Le principe d'une méthode de descente consiste à faire les itérations suivantes :

$$x_{k+1} = x_k + \lambda_k d_k, \qquad k > 0 \tag{2.8}$$

tout en assurant la propriété

$$f(x_{k+1}) < f(x_k)\ .$$

Le vecteur d_k est la direction de descente en x_k . Le scalaire λ_k est appelé le pas de la méthode à l'itération k.
On peut caractériser les directions de descente en x_k à l'aide du gradient :

Proposition 2.1 *Soit $d \in \mathbb{R}^n$ vérifiant*

$$\nabla f(x)^t.d < 0$$

alors d est une direction de descente en x.

Preuve. on a pour $\lambda > 0$

$$f(x + \lambda d) = f(x) + \lambda \nabla f(x)^t d + \lambda \varepsilon(\lambda)$$

donc si on écrit

$$\frac{f(x + \lambda d) - f(x)}{\lambda} = \nabla f(x)^t d + \varepsilon(\lambda)$$

on voit bien que pour λ suffisamment petit on aura

$$f(x + \lambda d) - f(x) < 0.$$

■

Ou encore que d fait avec l'opposé du gradient $-\nabla f(x)$ un angle strictement plus petit que $90°$:

$$\theta := arccos \frac{-\nabla f(x)^t d}{\|\nabla f(x)\| \, \|d\|} \in \left]0, \frac{\pi}{2}\right[$$

L'ensemble des directions de descente de f en x : $\{d \in \mathbb{R}^n : \nabla f(x)d < 0\}$
forme un demi-espace ouvert de $\mathbb{R}^n$ (illustration à la figure 2.1) .

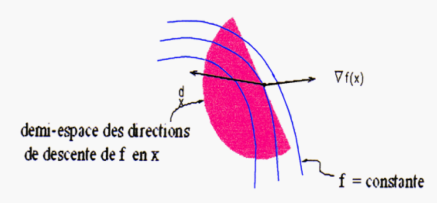


Fig.2.1 Demi-espace des
direction de descente

De telles directions sont intéressantes en optimisation car pour faire décroître f, il suffit de faire un déplacement le long de d.
Les méthodes à directions de descentes utilisent cette idée pour minimiser une fonction Dans la méthode (2.8) le choix de λ_k est lié à la fonction : $\varphi(\lambda) = f(x_k + \lambda d_k)$
Comme dans la méthode de la direction de descente, la trajectoire de la solution suit un modèle de zigzag . Si est choisi tels que $f(x_k + d_k)$ soit le minimum dans chaque itération, alors les directions successives sont orthogonales .
En effet
si on note $g(x) = \nabla f(x)$

$$\begin{aligned} \frac{df(x_k + \lambda d_k)}{d\lambda} &= \sum_{i=1}^{n} \frac{\partial f(x_k + \lambda d_k)}{\partial x_{ki}} \frac{d(x_{ki} + \lambda d_{ki})}{d\lambda} \\ &= \sum_{i=1}^{n} g_i(x_k + \lambda d_k) d_{ki} = g(x_k + \lambda d_k)^t d_k \end{aligned}$$

où $g(x_k + d_k)$ est le gradient au point $x_k + d_k$.
En particulier, une façon de choisir λ_k peut être de résoudre le problème d'optimisation (à une seule variable)

$$\min_{\lambda > 0} \varphi(\lambda) . \tag{2.9}$$

Si le pas $\tilde{\lambda}_k$ obtenu ainsi s'appelle le pas optimal alors nous pouvons écrire :

$$\varphi'\left(\tilde{\lambda}_k\right) = \nabla f(x_k + \tilde{\lambda}_k d_k)^t d_k = 0$$

c'est- à-dire $g(x_k + \tilde{\lambda}_k d_k)^t d_k = 0$
ou bien $d_{k+1}^t d_k = 0$

où

$$d_{k+1} = -g(x_k + \tilde{\lambda}_k d_k) = -g_{k+1}$$

est la direction de descente au point $x_k + \tilde{\lambda}_k d_k$. Donc les directions successives d_k et d_{k+1} sont orthogonaux comme représenté dans la figure (2.2) .

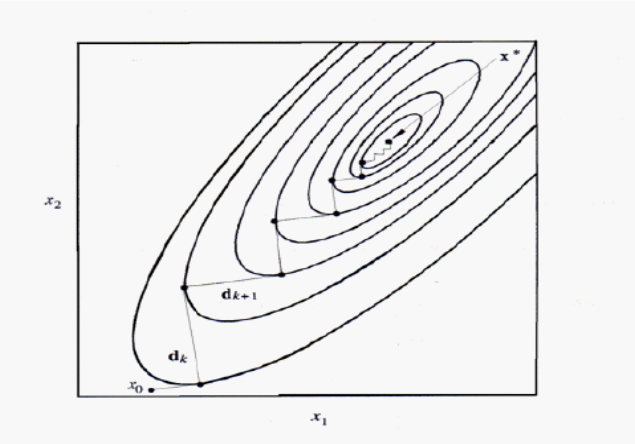


Fig .2.2 Trajectoire d'une solution typique d'une méthode à direction de déscente.

Pour définir une direction de descente il faut donc spécifier deux choses :

* Dire comment la direction d_k est calculée . Ce choix influe directement dans la nomination de l'algorithme.
* Dire comment on détermine le pas λ_k, c'est ce que l'on appelle : la recherche linéaire .

Algorithme 1.1 (méthode à directions de descente - une itération)

Etape 0 : (initialisation)

On suppose qu'au début de l'itération k, on dispose d'un itéré $x_k \in \mathbb{R}^n$.

Etape 1 :

Test d'arrêt : si $\|\nabla f(x_k)\| \simeq 0$, d'arrêt de l'algorithme .

Etape 2 :

Choix d'une direction de descente $d_k \in \mathbb{R}^n$.

Etape 3 :

Recherche linéaire : déterminer un pas $\lambda_k > 0$ le long de d_k de manière à "faire décroitre f suffisamment" .

Etape 4 :

Si la recherche linéaire est finie $x_{k+1} = x_k + \lambda_k d_k$, remplacer k par $k+1$ et aller à l'étape 1.

2.4.2 Recherche linéaire

Faire la recherche linéaire veut dire résoudre le problème unidimensionnel (2.9), où l'objectif est de :

* Faire décroître f suffisamment, cela se traduit le plus souvent par la réalisation d'une inégalité de la forme

$$f(x_k + \lambda_k d_k) \leq f(x_k) + \text{"un terme négatif "} \tag{2.10}$$

Le terme négatif, disons ν_k, joue un rôle-clé dans la convergence de l'algorithme utilisant cette recherche linéaire.

L'argument est le suivant .

Si $f(x_k)$ est minorée ($\exists c$ telleque $f(x_k) \geq c$ pour tout k), alors ν_k tend nécessairement vers zéro ($\nu_k \to 0$). c'est souvent à partir de la convergence vers zéro de cette suite que l'on parvient à montrer que le gradient lui-même doit tendre vers zéro. Le terme négatif devra prendre une forme bien particulière si on veut pouvoir en tirer de l'information .

En particulier, il ne suffit pas d'imposer $f(x_k + \lambda_k d_k) < f(x_k)$.

* Empêcher le pas $\lambda_k > 0$ d'être trop petit, trop proche de zéro.

Le premier objectif n'est en effet pas suffisant car l'inégalité (2.10) est en général satisfaite par des pas $\lambda_k > 0$ arbitrairement petit .
Or ceci peut entraîner une "fausse convergence", c'est-à-dire la convergence des itérés vers un point non stationnaire.
On donne dans cette partie un aperçu sur les recherches linéaires qu'on utilisera plus tard.On les a classées en deux catégories :

2.4.3 Les recherches linéaires exactes

Dans ce cas la solution optimale λ_k est calculée de façon exacte (d'un point de vue théorique car pratiquement on n'obtient
en général qu'une approximation).
On donnera l'algorithme de la recherche linéaire de dichotomie et du nombre d'or .
L'intervalle d'incertitude

Définition 2.3 *Considérons le problème unidimensionnel suivant* :

$$\underset{\lambda \in [a,b]}{\textit{Minimiser}}\ \varphi(\lambda)\ .$$

L'intervalle $[a,\ b]$ *est dit intervalle d'incertitude si le minimum* $\bar{\lambda}$ *de* $\varphi(\lambda)$ *appartient à* $[a,\ b]$*, mais sa valeur exacte n'est pas connue .*

Théorème 2.5 *soit* $\varphi : \mathbb{R} \to \mathbb{R}$ *strictement quasi-convexe sur* $[a,\ b]$.
soient $\lambda,\ \mu \in]a,\ b[\ ,\ \lambda < \mu$

1) *si* $\varphi\ (\lambda) > \varphi(\mu)$ *, alors* $\varphi\,(z) \geq \varphi(\mu); \forall z \in [a,\ \lambda]$.

2) *si* $\varphi\ (\lambda) \leq \varphi(\mu)$ *, alors* $\varphi\,(z) \geq \varphi(\lambda); \forall z \in [\mu,\ b]$.

Conséconces importante du théorème 2.5 :

1. Si $\varphi\ (\lambda) > \varphi(\mu)$, alors le nouveau intervalle d'incertitude est : $[\mu, b]$ (On supprime $[a, \lambda[$) .
2. Si $\varphi\ (\lambda) \leq \varphi(\mu)$, alors le nouveau intervalle d'incertitude est : $[a, \lambda]$ (On supprime $[\mu, b[$) .

Ceci est l'idée de base pour la construction d'algorithmes d'optimisation unidimentionnelle sans calcul de dérivées. A chaque itération on fait diminuer l'intervalle d'incertitude jusqu'à ce qu'on arrive à un intervalle final de longeur inferieur à une tolérance fixée à l'avance. Bien sur une valeur quelconque de ce dernier inntervalle conviendrait comme approximation de notre solution optimale.
Maintenant on va présenter deux méthodes d'optimisation unidimensionnelle sans dérivées :

La méthode de dichotomie

Algorithme de la méthode de dichotomie :
initialisation : Choisir $\varepsilon > 0$ et l longueur final de l'intervalle d'incertitude,$[a_1, b_1]$ étant l'intervalle initial.
poser $k = 1$ et aller à l'étape 1.
Etape1 **:**Si $b_k - a_k$ <l stop.Le minimum appartient à $[a_k; b_k]$.
Sinon poser :

$$\begin{aligned} \lambda_k &= \frac{a_k + b_k}{2} - \varepsilon \\ \mu_k &= \frac{a_k + b_k}{2} + \varepsilon \end{aligned}$$

et aller à l'étape2.
Etape2 **:**Si $\varphi(\lambda_k) > \varphi(\mu_k)$ alors $a_{k+1} = a_k, b_{k+1} = \mu_k$.
Sinon $a_{k+1} = \lambda_k, b_{k+1} = b_k$.
Remplacer k par $k + 1$ et allez à l'étape1.

La méthode du nombre d'or

La méthode du nombre d'or améliore la méthode de dichotomie, en diminuant le nombre d'observations, à chaque itération .

Algorithme de la méthode du nombre d'or :

Etape initiale :choisir $l > 0$ longueur final de l'intervalle d'incertitude et $[a_1, b_1], \alpha = 0,618$,calculer $\lambda_1$et μ_1 telle que :

$$\begin{aligned} \lambda_1 &= a_1 + (1-\alpha)(b_1 - a_1). \\ \mu_1 &= a_1 + \alpha(b_1 - a_1). \end{aligned}$$

Poser $k = 1$ et aller à Etape principale.

Etape principale :

(1) Si $b_k - a_k < l$ stop, prendre $\alpha^* \in [a_k, b_k]$. Si $\varphi(\lambda_k) > \varphi(\mu_k)$ aller à (2), sinon aller à (3).

(2) Poser $a_{k+1} = \lambda_k$, $b_{k+1} = b_k$,$\lambda_{k+1} = \mu_k$, $\mu_{k+1} = a_{k+1} + (b_{k+1} - a_{k+1})$, calculer $\varphi(\mu_{k+1})$, et aller à (4) .

(3) Poser $a_{k+1} = a_k$, $b_{k+1} = \mu_k$,$\mu_{k+1} = \lambda_k$, $\lambda_{k+1} = a_{k+1} + (1-\alpha)(b_{k+1} - a_{k+1})$, calculer$\varphi(\lambda_{k+1})$et aller à (4) .

(4) Poser $k = k + 1$, et aller à (1).

2.4.4 Les recherches linéaires inexactes

Les recherches linéaires exactes, malgré qu'elles n'aboutissent qu'à une solution optimale approchée, elle nécessitent boucoup d'observations à chaque itération de l'algorithme principal .Dans les années 60, 70, 80 ,des mathématiciens ont réusi à élaborer des recherches linéaire qui sont moins couteuse, mais respecte en même temps la descente de la fonction .

Décrivons maintenant en détail les trois recherches linéaires inexactes les plus importantes . Il s'agit des recherches linéaires inexactes d'Armijo, de Goldstein et de Wolfe .

Recherche linéaire inexacte d'Armijo (1966)

Soit $f : \mathbb{R}^n \to \mathbb{R}$, $x_k \in \mathbb{R}^n, d_k \in \mathbb{R}^n$ une direction de descente $(\nabla f(x_k)^t d_k < 0)$.

La règle d'Armijo exige à ce que f décroisse de façon suffisante au point $x_k + \lambda_k d_k$ .Cette condition est décrite par l'inégalité suivante appelée condition d'Armijo :

$$f(x_k + \lambda_k d_k) \leq f(x_k) + \varepsilon \lambda_k \nabla f(x_k)^t d_k, \qquad \varepsilon \in]0,1[\qquad \text{(Armijo)}$$

C'est à dire que la réduction de f doit être proportionnelle en même temps à λ_k et à la dérivée directionnelle $\nabla f(x_k)^t d_k$.

Interprétation graphique de la condition d'Armijo

Définissons la fonction

$$\varphi : \mathbb{R} \to \mathbb{R}$$

par

$$\varphi(\lambda_k) = f(x_k + \lambda_k d_k), \qquad \lambda_k \geq 0$$

Notons que :

$$\begin{aligned} \varphi'(\lambda) &= \nabla f(x_k + \lambda_k d_k)^t d_k, \\ \varphi'(0) &= \nabla f(x_k)^t d_k < 0, \\ \varphi(0) &= f(x_k). \end{aligned}$$

L'équation de la tangente au point $(0, \varphi(0))$ est la suivante :

$$\{\lambda, y\} : \; y = \varphi(0) + \varphi'(0)(\lambda - 0)$$

$$\tilde{\varphi}(\lambda_k) = f(x_k) + \nabla f(x_k)^t d_k \lambda_k$$

Posons

$$\tilde{\varphi}(\lambda) = \varphi(0) + \varphi'(0)\lambda.$$

L'équation de la tangente devient :

$$\tilde{\varphi}(\lambda) = f(x_k) + \nabla f(x_k)^t d_k \lambda$$

Définissons maintenant la fonction $\hat{\varphi}(\lambda)$ comme suit :

$$\hat{\varphi}(\lambda) = \varphi(0) + \varepsilon\lambda\varphi'(0) = f(x_k) + \varepsilon\lambda\nabla f(x_k)^t d_k, \qquad \varepsilon \in]0, 1[\tag{2.11}$$

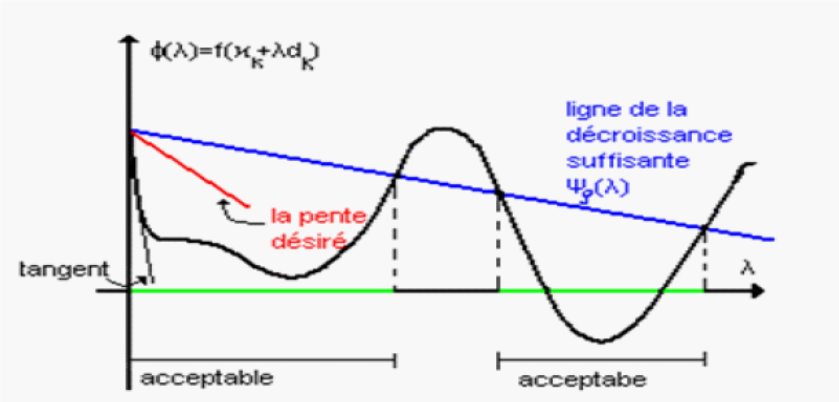


Fig.2.3– Règle d'Armijo

On cherche $\bar{\lambda}_k$ tel que

$$\varphi\left(\bar{\lambda}_k\right) \leq \hat{\varphi}\left(\bar{\lambda}_k\right).$$

Remarque 2.1 **1-** *La condition* $\varphi\left(\bar{\lambda}_k\right) \leq \hat{\varphi}\left(\bar{\lambda}_k\right)$ *implique la décroissance de la fonction* f.

En effet

$$\varphi\left(\bar{\lambda}_k\right) \leq \hat{\varphi}\left(\bar{\lambda}_k\right)$$

$$f\left(x_k + \bar{\lambda}_k d_k\right) \leq f\left(x_k\right) + \epsilon\bar{\lambda}_k \nabla f\left(x_k\right) d_k < f\left(x_k\right)$$

car la direction d *est une direction de descente.*

2- *Quand on prend* $\bar{\lambda}_k$ *très proche de zéro cela va nuire à la convergence et la vitesse de convergence.*

En effet

$$f\left(x_k + \bar{\lambda}_k d_k\right) = f(x_k) + \bar{\lambda}_k \nabla f(x_k) d_k + \bar{\lambda}_k \alpha(x_k, \bar{\lambda}_k d_k)$$

$$f\left(x_k + \bar{\lambda}_k d_k\right) - f(x_k) = \bar{\lambda}_k \left[\nabla f(x_k) d_k + \alpha(x_k, \bar{\lambda}_k d_k)\right]$$

$$\textit{si } \bar{\lambda}_k \to 0 \quad \underset{\bar{\lambda}_k \to 0}{\alpha(x_k, \bar{\lambda}_k d_k) \to 0} \textit{ donc} \quad f\left(x_k + \bar{\lambda}_k d_k\right) \simeq f(x_k).$$

Algorithme (Règle d'Armijo)

Etape 0 :(initialisation)

$\alpha_{g,1} = \alpha_{d,1} = 0$, choisir $\alpha_1 > 0, \rho \in]0,1[$ poser $k = 1$ et aller à l'étape 1.

Etape 1 :

si $\varphi_k(\alpha_k) \leq \varphi_k(0) + \rho\varphi_k'(0)\alpha_k$: STOP $(\alpha^* = \alpha_k)$.

si $\varphi_k(\alpha_k) > \varphi_k(0) + \rho\varphi_k'(0)\alpha_k$, alors

$\alpha_{d,k+1} = \alpha_d, \alpha_{g,k+1} = \alpha_k$ et aller à l'étape 2.

Etape 2 :

si $\alpha_{d,k+1} = 0$ déterminer $\alpha_{k+1} \in]\alpha_{g,k+1}, +\infty[$

si $\alpha_{d,k+1} \neq 0$ déterminer $\alpha_{k+1} \in]\alpha_{g,k+1}, \alpha_{d,k+1}[$

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

Remarque 2.2 *Il est clair d'après la figure Fig* 2.3*–Règle d'Armijo que l'inégalité d'Armijo est toujours vérifiée si* :

$\alpha_k \succ 0$ *est suffisamment petit . en effet, dans le cas contraire, on aurait une suite de pas strictement positifs* $\{\alpha_{k,i}\}_{i\geq 1}$ *convergeant vers* 0 *lorsque* $i \to \infty$ *et tels que* $f(x_k + \alpha_k d_k) \leq f(x_k) + \rho\alpha_k \nabla^T f(x_k) d_k$ *n'ait pas lieu pour* $\alpha_k = \alpha_{k,i}$.

En retranchant $f(x_k)$ *dans les deux membres, en divisant par* $\alpha_{k,i}$ *et en passant à la limite quand* $i \to \infty$, *on trouverait*

$$\nabla^T f(x_k) d_k \geq \rho \nabla^T f(x_k) d_k$$

ce qui contredirait le fait que d_k *est une direction de descente* $(\rho < 1)$.

Théorème 2.6 *Si* $\varphi_k : \mathbb{R}_+ \to \mathbb{R}$, *définie par* $\varphi_k(\alpha) = f(x_k + \alpha d_k)$ *est continue et bornée inférieurement, si* d_k *est une direction de descente en* x_k $(\varphi_k'(0) < 0)$ *et si* $\rho \in]0,1[$, *alors l'ensemble des pas vérifiant la règle d'Armijo est non vide* .

Preuve. *On a*

$$\varphi_k(\alpha) = f(x_k + \alpha d_k)$$

$$\Psi_\rho(\alpha) = f(x_k) + \rho\alpha_k \nabla^T f(x_k) d_k$$

Le developpement de Taylor-Yong en $\alpha = 0$ *de* φ_k *est* :

$$\varphi_k(\alpha) = f(x_k + \alpha d_k) = f(x_k) + \rho\alpha_k \nabla^T f(x_k) d_k + \alpha\xi(\alpha)$$

où $\xi(\alpha) \to 0, \alpha \to 0$
et comme $\rho \in]0,1[$ *et* $\varphi_k'(0) = \nabla^T f(x_k) d_k < 0$ *on déduit* :

$$f(x_k) + \alpha_k \nabla^T f(x_k) d_k < f(x_k) + \rho\alpha_k \nabla^T f(x_k) d_k, \text{pour } \alpha > 0$$

On voit que pour$\alpha > 0$ *assez petit on a* :

$$\varphi_k(\alpha) < \Psi_\rho(\alpha) \ .$$

De ce qui précède et du fait que φ_k *est bornée inférieurement, et* $\Psi_\rho(\alpha) \to -\infty, \alpha \to +\infty$, *on déduit que la fonction* $\Psi_\rho(\alpha) - \varphi_k(\alpha)$ *à la propriété* :

$$\begin{cases} \Psi_\rho(\alpha) - \varphi_k(\alpha) \succ 0 \text{ pour } \alpha \text{ assez petit} \\ \Psi_\rho(\alpha) - \varphi_k(\alpha) \prec 0 \text{ pour } \alpha \text{ assez grand} \end{cases}$$

donc s'annule au moins une fois pour $\alpha > 0$:
En choisissant le plus petit de ces zéros on voit qu'il existe $\bar{\alpha} > 0$ *tel que*

$$\varphi_k(\bar{\alpha}) = \Psi_\rho(\bar{\alpha}) \text{ et } \varphi_k(\alpha) < \Psi_\rho(\alpha) \text{ pour } 0 < \alpha < \bar{\alpha}.$$

Ce qui achève la démonstration . ■

Recherche linéaire inexacte de Goldstein (1967)

Le pas λ_k est acceptable par la recherche linéaire inexacte de Goldestein, s'il satisfait les deux conditions Goldestein1et Goldestein2 suivantes :

$$f(x_k + \lambda_k d_k) \leq f(x_k) + c\lambda_k \nabla f(x_k)^t . d_k \qquad c \in \left]0, \frac{1}{2}\right[\qquad \text{(Goldstein1)}$$

$$f(x_k + \lambda_k d_k) \geq f(x_k) + (1-c).\lambda_k.\nabla f(x_k)^t.d_k \qquad \text{(Goldstein2)}$$

Interprétation de la relation Goldstein1 :
La condition Goldstein1 est exactement la condition d'Armijo étudiée précédemment .Cette condition assure une décroissance suffisante de la fonction f.

Interprétation de la relation Goldstein2 :
La condition Goldstein2 évite au pas λ_k d'être trop petit (voir la figure ci dessous) .Ceci est d'un grand apport dans le processus de la convergence .

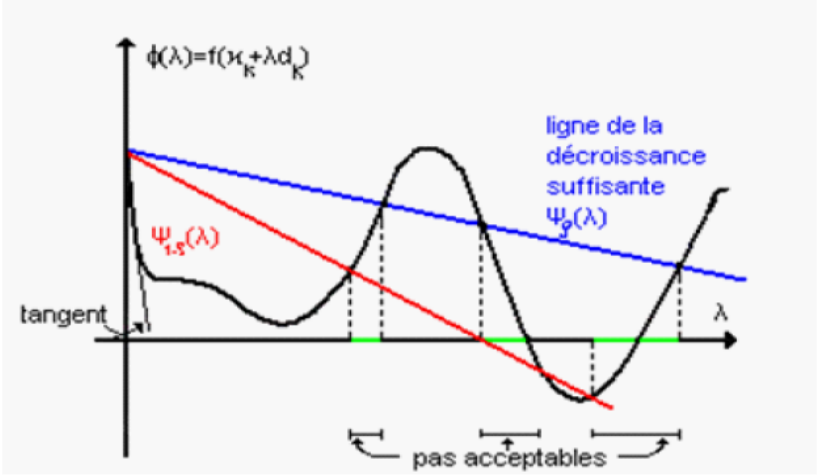


Fig.2.4– Règle de Goldstein

La figure 2.4 montre, sur un exemple, l'ensembles des points satisfaisant les deux conditions Goldstein .
L' Algorithme de Goldstein :
L' Algorithme essaye de trouver $\lambda_k \in]\beta_1, \beta_2[$. On démarre avec un intervalle $[a_0, b_0]$ assez grand . On prend $\lambda_0 \in]\beta_1, \beta_2[$:
Si λ_0 vérifié Goldstein1 et Goldstein2 alors $\lambda_0 \in]\beta_1, \beta_2[$ et on s'arrête .
Si $\lambda_0 > \beta_2$ alors λ_0 ne vérifié pas Goldstein, alors on prend $b_1 = \lambda_0$ et $a_1 = b_0$ et $\lambda_1 = \frac{a_1+b_1}{2}$ et on recommence avec λ_1.
Si $\lambda_0 < \beta_1$ alors λ_0 ne vérifié pas Goldstein2, on prend $a_1 = \lambda_0, b_1 = b_0$ et $\lambda_1 = \frac{a_1+b_1}{2}$ et on teste de nouveau λ_1 .
A l'itération k
Supposons qu'on ait $[a_k, b_k]$ et $\lambda_k = \frac{a_k+b_k}{2}$

Si λ_k vérifié Goldstein1 et Goldstein2, $\lambda_k \in]\beta_1, \beta_2[$. Stop .
Si λ_k ne vérifié pas Goldstein1 alors $\lambda_k > \beta_2$
On prend $b_{k+1} = \lambda_k$, $a_{k+1} = a_k$, $\lambda_{k+1} = \frac{a_{k+1}+b_{k+1}}{2}$.
Si λ_k ne vérifié pas Goldstein2 alors $\lambda_k < \beta_1$. On prend $a_{k+1} = \lambda_k$, $b_{k+1} = b_k$, $\lambda_{k+1} = \frac{a_{k+1}+b_{k+1}}{2}$.
On obtient ainsi l'algorithme suivant :

Algorithme de Goldstein

ETAPE 1 (Initialization)

Choisir $\alpha_0 \in [0, 10^{100}]$ et $\rho \in]0, 1[$. Poser $a_0 = 0, b_0 = 10^{100}$
Poser $k = 0$ et aller à ETAPE2

ETAPE2 (Test Goldstein1)

Itération k on a $[a_k, b_k]$ et α_k, calculez $\varphi_k(\alpha_k)$
Si $\varphi_k(\alpha_k) \leq \varphi_k(0) + \rho\alpha_k\varphi'_k(0)$, allez à ETAPE 3
Sinon
Poser $b_{k+1} = \alpha_k, a_{k+1} = a_k$, et allez à ETAPE 4

ETAPE 3 (Test Gold 02)

Si $\varphi_k(\alpha_k) \geq \varphi_k(0) + (1-\rho)\alpha_k\varphi'_k(0)$, stop. $\alpha^* = \alpha_k$
Sinon
Poser $a_{k+1} = \alpha_k$, $b_{k+1} = b_k$ et allez à ETAPE 4

ETAPE 4

Poser $\alpha_{k+1} = \frac{a_{k+1}+b_{k+1}}{2}$
Poser $k = k + 1$ et allez à ETAPE 2 .

Recherche linéaire inexacte de Wolfe (1969)

Recherche linéaire inexacte de Wolfe faible

Le pas λ_k est acceptable par la recherche linéaire inexacte de Wolfe faible ou de Wolfe tout simplement, s'il satisfait les deux conditions suivantes :

$$f(x_k + \lambda_k d_k) \leq f(x_k) + c_1\lambda_k\nabla^t f(x_k)d_k, \quad c_1 \in]0, 1[\qquad \text{(Wolfe1)}$$

$$\nabla f(x_k + \lambda_k d_k)^t d_k \geq c_2\nabla f(x_k)^t d_k, \quad c_2 \in]c_1, 1[\qquad \text{(Wolfe2)}$$

Interprétation de la relation Wolfe1

La condition Wolfe1 est exactement la condition d'Armijo, cette condition assure une décroissance suffisante de la fonction f .

Interprétation de la relation Wolfe2

Les λ_k séléctionnés par la condition Wolfe1 peuvent être trés petits . Ceci peut avoir des conséquences facheuses sur la convergence de l'algorithme. La condition Wolfe2 évite cet inconvénient et supprime les très petites valeurs de λ_k. (voir la fugire ci dessous) .

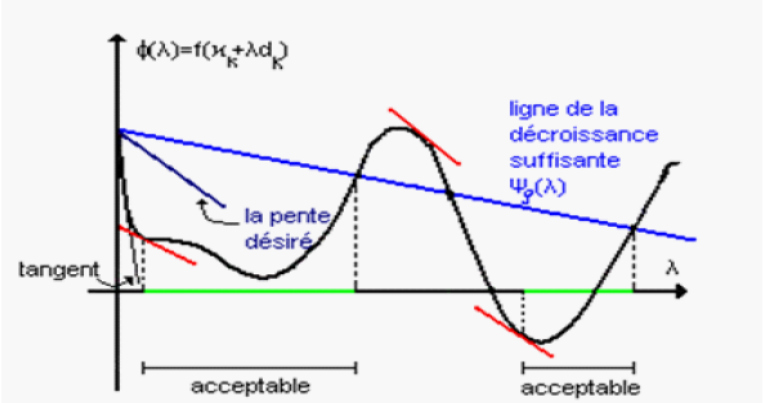


Fig.2.5– Règle de Wolfe

Le figure 2.5 montre sur un exemple l'ensembles des points satisfaissant les conditions de Wolfe $c_1 = 0,1$; $c_2 = 0,7$ (Lemaréchal 1980).

Recherche linéaire inexacte de Wolfe forte

Le pas λ_k est acceptable par la recherche linéaire inexacte de Wolfe forte, s'il satisfait les deux conditions Wolfe forte1 et Wolfe forte2 suivantes :

$$f(x_k + \lambda_k d_k) \leq f(x_k) + c_1 \lambda_k \nabla f(x_k)^t d_k, \quad c_1 \in]0,1[\qquad \text{(Wolfe forte1)}$$

$$\left|\nabla f(x_k + \lambda_k d_k)^t . d_k\right| \leq c_2 . \left|\nabla f(x_k)^t . d_k\right|, \quad c_2 \in]c_1, 1[\qquad \text{(Wolfe forte2)}$$

Interprétation de la relation Wolfe forte1

La condition Wolfe forte1 est exactement la condition Wolfe1 ou Armijo. Cette condition assure une décroissance suffisante de la fonction f.

Interprétation de la relation Wolfe forte2

La condition Wolfe forte2 implique Wolfe2. Le pas λ_k sélectionné par les condition Wolfe1 et Wolfe2 peut être très loin d'un point optimal ou stationnaire de la fonction φ . La condition Wolfe forte2 assure que le pas λ_k se trouve dans le voisinage d'un point stationnaire ou un point optimal de φ .

L'algoritheme de wolfe

ETAPE 1 (Initialisation)

Prendre $\alpha_0 \in [0, 10^{99}]$, calculez $\varphi(0), \varphi'_0(0)$. Prendre $\rho = 0.1$ (ou $\rho = 0,1$ ou $\rho = 0,001$ ou $\rho = 10^{-4}$) $\theta = 0.9$ (ou plus petit encore)

Poser $a_0 = 0, b_0 = 10^{99}, k = 0$ et allez à ETAPE 2

ETAPE 2 (test de (Wolfe1))

Calculez $\varphi(\alpha_k)$. Si $\varphi(\alpha_k) \leq \varphi(0) + \rho\alpha_k\varphi'(0)$, aller à ETAPE 3. Sinon

Poser $a_{k+1} = a_k, b_{k+1} = \alpha_k$ et allez à ETAPE 4

ETAPE 3 (test (Wolfe2) ou (Wolfeforte2))

Calculez $\varphi'(\alpha_k)$. Si $\varphi'(\alpha_k) \geq \theta\varphi'(0)(|\ \varphi'_k(\alpha_k)| \leq -\theta\varphi'(0))$. STOP

Prendre $\bar{\alpha} = \alpha_k$. Sinon Poser $a_{k+1} = \alpha_k, b_{k+1} = b_k$ et allez à ETAPE 4

ETAPE 4 (calcul de α_{k+1})

$\alpha_{k+1} = \frac{a_{k+1}+b_{k+1}}{2}$

Poser $k = k + 1$ et allez à ETAPE 2 .

2.4.5 Convergence des méthodes utilisant des recherches linéaires inexactes et des directions de descente. Le théorème de Zoudentijk

La condition de Zoutendijk

Maintenant on va étudier la contribution de la recherche linéaire inexacte dans la convergence des algorithmes à directions de descente. Ce n'est qu' une contribution, parce que la recherche linéaire ne peut à elle seule assurer la convergence des itérés . On comprend bien que le choix de la direction de descente joue aussi un rôle. Cela se traduit par une condition, dite de Zoutendijk, dont on peut tirer quelques informations qualitatives intéressantes.

On dit qu'une règle de recherche linéaire inexacte satisfait la condition de Zoutendijk s'il existe une constante $C > 0$ telle que pour tout indice $k \geq 1$ on ait

$$f(x_{k+1}) \leq f(x_k) - C \left\|\nabla f(x_k)\right\|^2 \cos^2 \theta_k \tag{2.12}$$

où θ_k est l'angle que fait d_k avec $-\nabla f(x_k)$, défini par

$$\cos \theta_k = \frac{-\nabla^T f(x_k) d_k}{\|d_k\| \, \|d_k\|}. \tag{2.13}$$

Voici comment on se sert de la condition de condition de Zoutendijk.

Théorème 2.7 (***de Zoutendijk***)

Si la suite $\{x_k\}$ générée par un algorithme d'optimisation vérifie la condition de Zoutendijk (2.12) *et si la suite $\{f\,(x_k)\}$ est minorée, alors*

$$\sum_{k\geq 1} \left\|\nabla f(x_k)\right\|^2 \cos^2 \theta_k < \infty \tag{2.14}$$

Preuve. *En sommant les quantités inégalités* $\|\nabla f(x_k)\|^2 \cos^2\theta_k$ *tout en prenant en considération* (2.13) *, on a*

$$\sum_{k\geq 1}^{l} \|\nabla f(x_k)\|^2 \cos^2\theta_k \leq \frac{1}{C}\left(f(x_1) - f(x_{l+1})\right)$$

La série est donc convergente puisqu'il existe une constant C' *telle que pour tout* k, $f(x_k) \geq C'$. ■

Conséquence importante du théorème de Zoudentijk

La condition (2.14) implique

$$\|\nabla f(x_k)\|^2 \cos^2\theta_k \to 0 \quad (k \to \infty) \tag{2.15}$$

Cette limite peut être utilisée pour en déduire la convergence de l'algorithme.
En effet si notre algorithme génère une suite $\{x_k\}$ de la forme :

$$x_{k+1} = x_k + \lambda_k d_k \ .$$

Si le choix de d_k est tel que

$$\cos\theta_k \geq \delta > 0, \quad \forall k$$

alors il découle de (2.15) que

$$\lim \| \nabla f(x_k) \| = 0 \tag{2.16}$$

Les deux propositions suivantes précisent les circonstances dans lesquelles la condition de Zoutendijk (2.12) est vérifiée avec les règles d'Armijo et de Wolfe.

Proposition 2.2 *Soit* $f : \mathbb{R}^n \to \mathbb{R}$ *une fonction continument différenciable dans un voisinage de* $T = \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$.
On considère un algorithme à directions de descente d_k, *qui génère une suite* $\{x_k\}$ *en utilisant la recherche linéaire d'Armijo, avec* $\alpha_1 > 0$.
Alors il existe une constante $C > 0$ *telle que, pour tout* $k \geq 1$, *l'une des conditions*

$$f(x_{k+1}) \leq f(x_k) - C\nabla^T f(x_k) d_k$$

ou

$$f(x_{k+1}) \leq f(x_k) - C \| \nabla f(x_k) \|^2 \cos^2\theta_k$$

est vérifiée.

Proposition 2.3 *Soit* $f : \mathbb{R}^n \to \mathbb{R}$ *une fonction continument différenciable dans un voisinage de* $T = \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$.

On considère un algorithme à directions de descente d_k, *qui génère une suite* $\{x_k\}$ *en utilisant la recherche linéaire de Wolfe* $(Wolfe1)et\ (Wolfe2)$.
Alors il existe une constante $C > 0$ *telle que, pour tout* $k \geq 1$, *la condition de Zoutendijk* (2.12) *est vérifiée.*

2.5 Convergence des algorithmes et fonctions multivoques

Les méthodes de résolution d'optimisation sont en général des méthodes itératives, c'est à dire à partir d'un point initial x_0, elles engendrent une suite infinie $x_1, x_2, ..., x_k, ...$ dont on espère que cette suite converge vers la solution optimale .

Définition 2.4 *Un algorithme de résolution est un procédé itératif qui permet d'un point initial* x_0 *d'engendrer la suite* $x_0, x_1, ..., x_k, ...$
Un algorithme est parfaitement définir par la donnée de l'application a *qui à* x_k *associe* $x_{k+1} = a(x_k)$ *. L'étude de la convergence de l'algorithme se ramène à l'étude des propriétés de* a .
Par exemple :

$$a :\longrightarrow a(x) = x - \lambda \nabla f(x)$$

a *représente la méthode du gradient.*

Un modèle général des algorithmes : application multivoque
Soit le problème (P) :

$$(P) : \min_{x \in S} f(x)$$

On peut générer plusieurs classes d'algorithmes qui ont pour but de résoudre le problème (P) ou d'atteindre des points vérifiant les conditions nécessaires d'optimalité. Il serait alors logique au lieu d'étudier la convergence d'un algorithme, il serait préférable d'établir une théorie ou un model assez général qui étudie la convergence d'une classe d'algorithme au lieu d'un seul algorithme.
Soient $a_1, a_2, a_3, ..., a_P$ des applications qui génèrent p algorithmes, c'est à dire qu'à un point x_k on associe les p points différents $a_1(x_k)$, $a_2(x_k)$, $a_3(x_k)$, ..., $a_p(x_k)$.
C'est à dire qu'on peut considérer l'application qui à x_k fait correspondre

$$A(x_k) = (a_1(x_k), a_2(x_k), a_3(x_k), ..., a_p(x_k)).$$

Ceci conduit naturellement à un modèle dans lequel les algorithmes sont représentés par des applications multivoques, c'est à dire des applications de $\mathbb{R}^n$ dans $P(\mathbb{R}^n)$ qui à $x \in \mathbb{R}^n$ fait correspondre une partie de $\mathbb{R}^n$.

D'une façon générale on notera :

$$A : X \overset{m}{\to} Y$$

une application multivoque de X dans Y.

Convergence globale :

Définition 2.5 *On dit qu'un algorithme décrit par une application multivoque A, est globalement convergent si quelque soit le point de départ x_0 choisi, la suite $\{x_k\}$ engendrée $x_{k+1} \in A(x_k)$ (ou une sous-suite) converge vers un point satisfaisant les conditions nécessaire d'optimalité (ou solution optimale) .*

Notation 2.1 :

$$\Omega = \left\{ \begin{array}{c} x : x \text{ satisfait une condition nécessaire d'optimalité} \\ \text{ou } x \text{ une solution optimale locale ou globale} \end{array} \right\}$$

Ω s'appelle ensemble des solutions .

Applications multivoques fermées :

C'est une généralisation de la notion de continuité pour les fonctions univoques .

Définition 2.6 *Soit $A : X \overset{m}{\to} Y$ une application multivoque.*
On dit que A est fermée en $x \in X$ si et seulement si :

$$\left. \begin{array}{c} \text{pour toute suite } \{x_k\}_{k\in\mathbb{N}},\ x_k \to x \text{ dans } X \\ \text{et pour toute suite } \{y_k\}_{k\in\mathbb{N}},\ y_k \to y \text{ dans } Y \end{array} \right\} \Longrightarrow y \in A(x)$$

A est dite fermé sur $S \subset X$ si et seulement si A est fermée en tout point $x \in S$.

Remarque 2.3 *Si A est univoque continue, alors A est fermée.*
En effet, soit $A : X \to Y$ continu en x et soit

$$\left\{ \begin{array}{c} x_k \to x \text{ dans } X \\ A(x_k) = y_k \to y \text{ dans } Y \end{array} \right\} \Longrightarrow A(x) = y \text{ d'après la continuité de } A \text{ .}$$

Composition d'applications multivoques :

Définition 2.7 *Soient $A : X \overset{m}{\to} Y$ et $B : Y \overset{m}{\to} Z$ (X, Y, Z fermés non vides) .*
La composée $B \circ A$ est l'application multivoque $C : X \overset{m}{\to} Z$ définie par : $C(x) = \underset{y\in A(x)}{\cup} B(y)$.

Théorème 2.8 (*composition des applications multivoques fermées*) :

Soit $A : X \overset{m}{\to} Y$ *et* $B : Y \overset{m}{\to} Z$ *.On suppose que* :

1) A *est fermée en* $x \in X$ *et que* B *est fermée sur* $A(x)$.

2) *Toute suite* $\{y_k\}$ *telle que* $y_k \in A(x_k)$ *avec* $x_k \to x$, *admet une sous-suite convergente* .

Alors l'application multivoque $C = B \circ A$ *est fermée en* x.

Corollaire 2.1 *Soit* $A : X \to Y$ *une application univoque et* $B : Y \overset{m}{\to} Z$ *une application multivoque. Si* A *est continue en* x *et si* B *est fermée sur* $A(x)$, *alors* $B \circ A$ *est fermée en* x.

Preuve. On a A est continue en x, B est fermée sur $A(x)$ et on a $\{y_k\}$ tel que $y_k = A(x_k)$ avec $x_k \to x$ alors $A(x_k) \to A(x)$ donc $\{y_k\}$ est convergente, c'est à dire que les deux conditions du théorème 2.5 sont vérifiées, d'où $B \circ A$ est fermée . ■

Théorème de Zangwill :

Définition 2.8 *On dit que* $h : X \to \mathbb{R}$ *es une fonction de descente* (*relativement à une application multivoque* A) *si elle est continue et possède les propriétés suivantes* :

1- $x \notin \Omega \Longrightarrow h(y) < h(x)\,;\ (\forall y \in A(x))$.

2- $x \in \Omega \Longrightarrow h(y) \leqslant h(x)\,;\ (\forall y \in A(x))$.

Théorème 2.9 (***Zangwill* 1969**) :
Soit

$$(P) \ : \ \min_{x \in X} f(x)$$

un problème d'optimisation et Ω *l'ensemble des solutions*.
Soit $A : X \overset{m}{\to} X$ *une application multivoque et* $\{x_k\}$ *une suite engendrée par l'algorithme, c'est à dire vérifiant* $x_{k+1} \in A(x_k)$.
Supposons que les trois conditions suivantes soient vérifiées :

C1- *Les points* $\{x_k\}$ *sont tous contenus dans un ensemble compact* $K \subset X$.

C2- *Il existe une fonction de descente* h .

C3- *L'application multivoque* A *est fermée dans* $X \setminus \Omega$ *et* $\forall x \in X \setminus \Omega$, $A(x) \neq 0$.

Alors :
pour tout x *limite d'une sous-suite convergente de la suite* $\{x_k\}$, *on* $x \in \Omega$ *et* $h(x_k) \underset{\substack{k \to \infty \\ k \in \mathbb{N}}}{\to} h(x)$.

Les modes de convergence :
Notre but est d'accélérer la convergence d'une suite, donc il est naturel de présenter la notion de rapidité ou mode de convergence de suites.

Définition 2.9 *Soit* $\{x_k\}_{k\in\mathbb{N}}$ *une suite dans* $\mathbb{R}^n$ *qui converge vers* $\hat{x}$.

1- *On dit que* $\{x_k\}_{k\in\mathbb{N}}$ *converge vers* $\hat{x}$ *linéairement avec le taux* η *si*

$$\lim_{k\to\infty} \sup \frac{\|x_{k+1} - \hat{x}\|}{\|x_k - \hat{x}\|} = \eta < 1.$$

Lorsque $\|x_{k+1} - \hat{x}\| \simeq \eta \|x_k - \hat{x}\|$ *, la convergence est dite asymptotique .*

2- *Si*

$$\lim_{k\to\infty} \sup \frac{\|x_{k+1} - \hat{x}\|}{\|x_k - \hat{x}\|^\gamma} < +\infty,\ \gamma > 1$$

la convergence est dite super linéaire d'ordre γ*.*

3- *On dit que* $\{x_k\}_{k\in\mathbb{N}}$ *tend vers* $\hat{x}$ *de façon super linéaire si*

$$\lim_{k\to\infty} \frac{\| x_{k+1} - \hat{x} \|}{\| x_k - \hat{x} \|} = 0.$$

Lorsque $\|x_{k+1} - \hat{x}\| \leq l \|x_k - \hat{x}\|^\gamma$ *, la convergence est dite super linéaire asymptotiq.*

Chapitre 3

METHODE DE LA PLUS FORTE PENTE

La méthode du gradient fut découverte par Cauchy en 1847 est l'une des anciennes méthode les plus utilisées pour résoudre le problème (P), elle est également connu sous le nom de méthode de la plus forte pente ou de plus de la plus profonde descente (steepest descente, en englais) car 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 (voir Théorème 2.1) .
Le théorème suivant va montrer que la décroissance de la fonction sera la plus forte en suivant la direction $-\nabla f(x_k)$

Théorème 3.1 *Supposons que $f : \mathbb{R}^n \to \mathbb{R}$ soit différentiable au point x, et supposons que*

$$\nabla f(x) \neq 0$$

.
Considérons le problème optimal

$$\underset{\|d\| \leq 1}{Minimiser}\ f'(x,\ d)$$

où $f\prime(x,\ d)$ est la dérivée directionnelle de f au point x et dans la direction d. Alors la solution optimale de ce problème est donnée par

$$\tilde{d} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}.$$

Preuve. *On a*

$$f\prime(x, d) = \lim_{\lambda \to 0_+} \frac{f(x + \lambda d) - f(x)}{\lambda} = \nabla f(x)^t d$$

donc on va minimiser $\nabla f(x)^t d$ dans $\{d : \|d\| \leq 1\}$.
En appliquant l'inégalité de Schwartz, on obtient

$$\left|\nabla f(x)^t d\right| \leq \|\nabla f(x)\|\, \|d\| \tag{3.1}$$

Si

$$\nabla f(x)^t d \geq 0$$

alors

$$\nabla f(x)^t d \geq -\|\nabla f(x)\| \, \|d\|$$

Si

$$\nabla f(x)^t d \leq 0$$

(3.1) *implique*

$$-\nabla f(x)^t d \leq \|\nabla f(x)\| \, \|d\|$$

par conséquent on a toujours

$$\nabla f(x)^t d \geq -\|\nabla f(x)\| \, \|d\|$$

pour $\| d \| \leq 1$ *on a*

$$\|\nabla f(x)\| \, \|d\| \leq \|\nabla f(x)\| \Rightarrow -\|\nabla f(x)\| \, \|d\| \geq -\|\nabla f(x)\|$$

alors $\forall d : \|d\| \leq 1$ *on a*

$$\nabla f(x)^t d \geq -\|\nabla f(x)\|$$

D'autre part on a $\left\|\bar{d}\right\| = 1$ *et* $\bar{d}$ *vérifie*

$$\nabla f(x)^t \bar{d} = \nabla f(x)^t \left(\frac{\nabla f(x)}{\|\nabla f(x)\|} \right) = -\|\nabla f(x)\| \ .$$

■

Interprétation du théorème 3.1 : Nous allons à partir du théorème 3.1 donner une idée intuitive sur l'appellation : méthode de la plus forte pente . En effet d'après le théorème 3.1 on a :

$$f'(x,\ d) \geq f'(x,\ \bar{d}) \quad : \forall d, \|d\| \leq 1 \ .$$

En utilisant la définition de la dérivée directionnelle

$$\lim_{\lambda \to 0_+} \frac{f(x+\lambda d) - f(x)}{\lambda} \geq \lim_{\lambda \to 0_+} \frac{f(x+\lambda \bar{d}) - f(x)}{\lambda} \ .$$

Cette dernière inégalité implique qu'il existe $\delta > 0$ tel que :

$$[f(x+\lambda d) - f(x)] - \left[f(x+\lambda \bar{d}) - f(x)\right] \geq 0, \forall \lambda \in \ ]-\delta, +\delta[$$

alors

$$f(x+\lambda d) \geq f(x+\lambda \bar{d}), \forall \lambda \in \ ]-\delta, +\delta[\text{ et } \forall d : \|d\| \leq 1 \ .$$

3.1 Algorithme de la méthode de la plus forte pente

Etape initiale : Choisir un $\varepsilon > 0$. Choisir un point initial x_1. Poser $k = 1$ et aller à l'étape principale.
Etape principale : Si $le\nabla f(x)le < epsilon$ stop.
Sinon poser $d_k = -le\nabla f(x)le$ et soit λ_k la solution optimale de la recherche linéaire

$$\min \{f(x_k + \lambda d_k); \lambda \geq 0\}$$

Poser

$$x_{k+1} = x_k + \lambda_k d_k$$

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

3.2 Inconvénients de la méthode de la plus forte pente

Lenteur de la méthode au voisinage des points stationnaires :
Cette méthode travaille de façon performante dans les premiers étapes de l'algorithme mais au voisinage des points stationnaires elle devient très lente.
On peut expliquer ce problème comme suit :
On a

$$f(x_k + \lambda d) = f(x_k) + \lambda \nabla f(x_k)^t.d + \lambda \|d\| \alpha(x_k, \lambda d)$$

où $\alpha(x_k, \lambda d) \to 0$ quand $\lambda d \to 0$.
Si $d = -\nabla f(x_k)$ alors $x_{k+1} = x_k - \lambda \nabla f(x_k)$ et donc

$$f(x_k + \lambda d) - f(x_k) = \lambda \left[- \|\nabla f(x_k)\|^2 + \|\nabla f(x_k)\| \alpha(x_k, -\lambda \nabla f(x_k)) \right]$$

D'après l'expression précédent, on voit que lorsque x_k s'approche d'un point stationnaire, et si f est continument différentiable, alors $\|\nabla f(x_k)\|$ est proche de zéro . Donc le terme à droite s'approche de zéro, indépendamment de λ, et par conséquent $f(x_k+\lambda d)$ ne s'éloigne pas beaucoup de $f(x_k)$ quand en passe du point x_k au point x_{k+1}.
Le phénomène de Zigzaguing :
Il n'est pas facile de vérifier que pour la méthode du gradient on a toujours

$$d_k^t.d_{k+1} = 0$$

c'est à dire que la suite $\{x_k\}$ engendrée par l'algorithme de la méthode du gradient, zigzague . Ceci crée un phénomène de ralentissement dans l'acheminement des points x_k vers la solution optimale .

3.3 Quelques remèdes

Changement de direction :
Au lieu de prendre comme direction de descente, la direction :

$$d_k = -\nabla f(x_k)$$

on prend des directions de la forme :

$$d_k = -D.\nabla f(x_k)$$

où D est une matrice choisie convenablement (D pourrait être par exemple l'inverse de la matrice héssienne au point x_k c'est à dire $(H(x_k))^{-1}$) .
Un autre choix pourrait s'opérer de la façon suivante :

$$d_k = -\nabla f(x_k) + g_k$$

où g_k est un vecteur approprié .
Accélération de la convergence :
On peut aussi accélérer la convergence de la méthode du gradien. Pour cela on transforme grâce à un algorithme d'accélération de la convergence la suite $\{x_k\}$ en une suite $\{y_k\}$ qui convergerait vers la même limite que la suite $\{x_k\}$, mais convergerait plus rapidement . Si on note par x^* cette limite commune on exprime cette rapidité par la limite suivante :

$$\lim_{k\to\infty} \frac{y_k - x^*}{x_k - x^*} = 0.$$

3.4 Programme en fortran 90 de la méthode de la plus forte pente

Dans cette partie on présente le programme en Fortran 90 de l'algorithme steepest descente avec la recherche linéaire d'Armijo . On utilise le choix des paramètres suivants :
$\varepsilon = 10^{-5}, mu = 0.2, be = 0.5$
Programme en fortran 90
program steepest
implicit none
integer, parameter :: $n = 1000$
double precision $yc(n), pk(n)$

double precision $eps, s, be, f1, f0, mu, f2$
integer k,l,m,i
$k = 0; m = 0$
print*,'enter the initial point'
do i=1,n
$yc(i) = 1.$
enddo
print*,'enter eps'
read*,eps
$be = 0.5; mu = 0.2$
do
if $(sqrt$ (dot_product$(df(yc), df(yc))) <= eps$.or.$k >= 500)$ exit
$pk = -df(yc)$
$f0 = f(yc)$
!recherche d'Armijo
$l = 1; s = 1$
$f1 = f(yc + s^*pk)$
if $(f1 <= f0 + mu^*s^*dot\_product(df(yc), pk))$ then
$s = s/be$
do
$f2 = f(yc + s^*pk)$
if $(f2 > f0 + mu^*s^*dot\_product(df(yc), pk))$ exit
$s = s/be$
$f1 = f2; l = l + 1$
enddo
$s = s^*be$
$yc = yc + s^*pk$
$k = k + 1$
$print^*, s$
else
$s = s^*be$
do
$f1 = f(yc + s^*pk)$
if $(f1 <= f0 + mu^*s^*dot\_product(df(yc), pk))$ then
$yc = yc + s^*pk$

$k = k + 1$

$print^*, s$

exit

endif

$s = s^*be$

$l = l + 1$

enddo

endif

$m = m + l + 1$

$f0 = f1$

enddo

$print^*$,'LA SOLUTION EST'

$print$'$(2e10.2)$', yc

$print^*$,'LA VALEUR DE LA FONCTION EST'

$print$'$(e10.2)$',$f(yc)$

$print^*$,'LA NORME DU GRADIENT EST'

$print$ '$(e10.2)$',$sqrt(dot\_product(df(yc), df(yc)))$

$print^*$,'LE NOMBRE D IERATIONS EST'

$print$'$(i10)$',k

contains

function $f(u)$

double precision $f, u(n)$

integer $i1$

$f = 0$

do $i1 = 1, n$

$f = f + i1^*u(i1)^{**}2$

enddo

$f = f^{**}2$

endfunction

function $df(u)$

double precision $df(n), u(n), sm$

integer$i1$

$sm = 0$

do $i1 = 1, n$

$sm = sm + i1^*u(i1)^{**}2$

enddo
do $i1 = 1, n$
$df(i1) = 4^*i1^*u(i1)^*sm$
enddo
endfunction
end

3.5 Convergence de la méthode de la plus forte pente

3.5.1 Cas des recherches linéaires exactes

avant de donner le théorème principal concernant la convergence de la méthode de la plus forte pente avec des recherches linéaires exactes et inexactes, démontrons d'abord ce résultat qui nous sera utile dans la section qui va suivre .

Théorème 3.2 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ et $L \subset \mathbb{R}$, un intervalle fermé et M l'application multivoque suivante :*

$$M : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$$

$$(x, d) \to M(x, d) = \left\{ y : y = x + \bar{\lambda} d \right\}$$

$\bar{\lambda}$ vérifiant :

$$f(x + \bar{\lambda} d) = \min \left\{ f(x + \bar{\lambda} d); \lambda \in L \right\}.$$

Si f est continue en x et si $d \neq 0$, alors M est fermée au point (x, d) .

Preuve. *Soit $\{x_k, d_k\}_{k \in \mathbb{N}}$ et $\{y_k\}$ telles que*

$$(x_k, d_k) \underset{k \to \infty}{\to} (x, d), y_k \in M(x_k, d_k), y_k \underset{k \to \infty}{\to} y. \tag{3.2}$$

Démontrons que

$$y \in M(x, d).$$

En effet $y_k \in M(x_k, d_k)$. Alors

$$y_k = x_k + \bar{\lambda}_k . d_k \tag{3.3}$$

$\bar{\lambda}_k$ est solution optimale de la recherche linéaire

$$f\left(x_k + \bar{\lambda}_k . d_k\right) = \min \left\{ f\left(x_k + \lambda . d_k\right), \lambda \in L \right\} . \tag{3.4}$$

Puisque

$$d_k \underset{k \to \infty}{\to} d \neq 0$$

alors

$$d_k \neq 0, \quad \forall k \geq k_0, \quad k_0 \in \mathbb{N} \tag{3.5}$$

(3.3) *et* (3.5) *impliquent*

$$\bar{\lambda}_k = \frac{\|y_k - x_k\|}{\|d_k\|} \tag{3.6}$$

soit en remarquant que L *est fermé, on a*

$$\bar{\lambda}_k \underset{k\to\infty}{\to} \bar{\lambda} = \frac{\| y - x \|}{\| d \|} \in L \tag{3.7}$$

(3.2) (3.3) (3.6) *et* (3.7) *donnent*

$$y = x + \bar{\lambda}.d$$

(3.4) *implique*

$$f\left(x_k + \bar{\lambda}_k.d_k\right) \leq f\left(x_k + \lambda.d_k\right) \ .$$

Faisons tendre k *vers l'infini et prenons en considération le fait que f est continue au point* (x, d)*, on obtient* :

$$f\left(x_k + \bar{\lambda}_k.d_k\right) \leq f\left(x + \lambda.d\right), \quad \forall \lambda \in L.$$

Ceci veut dire exactement que le point $y = x + \bar{\lambda}.d \in M(x, d)$. ■

Théorème 3.3 (*convergence de la méthode de la plus forte pente avec des recherches linéaires exactes*) *Soit* $f : \mathbb{R}^n \to \mathbb{R}$*, telle que* $f \in C^1(\mathbb{R}^n)$ *et* (P) *le problème de minimisation sans contraintes suivant* :

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

On suppose que l'ensemble $\delta(x_0)$ *suivant* :

$$\delta(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0), \quad x_0 \in \mathbb{R}^n\}$$

est borné . Soit $\{x_k\}$ *une suite générée par l'algorithme de la plus forte pente, c'est à dire*
Initialisation : $x_0 \in \mathbb{R}^n$*, point initial, poser* $k = 0$ *et aller à Etape principale*
Etape principale : *Si* $\nabla f(x_k) = 0$ *Stop* .
Sinon
Calculer λ_k *vérifiant* $f(x_k + \lambda_k d_k) = min\left\{f(x_k + \lambda d_k), \lambda \geq 0\right\}$.
Calculer $x_{k+1} = x_k - \lambda_k \nabla f(x_k)$.
Poser $k = k + 1$ *et aller à Etape principale* .
Fin .
Soit x^* *une limite quelconque d'une sous suite convergente de* $\{x_k\}$ *. Alors* $\nabla f(x^*) = 0$.

3.5.2 Cas des recherches linéaires inexactes

Théorème 3.4 ([***Bazara***]) (*convergence de la méthode de la plus forte pente avec la recherche linéaire d'Armijo*) . *Soit* $f : \mathbb{R}^n \to \mathbb{R}$, *telle que* $\nabla f(x)$ *est continument Lipschitzien de constante* G *dans l'ensemble* $\delta(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$, $x_0 \in \mathbb{R}^n$ *quelconque . Considérons l'algorithme de la méthode de la plus forte pente avec la recherche linéaire d'Armijo, c'est à dire l'algorithme défini comme suit* :

Initialisation : *Fixons* $\bar{\lambda} > 0$, $0 < \varepsilon < 1$ *et* $x_0 \in \mathbb{R}^n$ *point initial, poser* $k = 0$ *et aller à Etape principale* .

Etape principale: *A l'itération* k *définissons la direction* $dk = -\nabla f(x_k)$ *et considérons la fonction d'Armijo*

$$\hat{\theta}(\lambda) = \theta(0) + \lambda\varepsilon\theta'(0) \tag{3.8}$$

où $\theta(\lambda)$ *est la fonction suivante* :

$$\theta(\lambda) = f(x_k + \lambda d_k) = f[x_k - \lambda\nabla f(x_k)] : \lambda \geq 0. \tag{3.9}$$

Si $\nabla f(x_k) = 0$ *stop* .

Sinon

Trouver le plus petit entier $t \geq 0$ *tel que*

$$\theta\left(\frac{\bar{\lambda}}{2^t}\right) \leq \hat{\theta}\left(\frac{\bar{\lambda}}{2^t}\right) \tag{3.10}$$

et définir le successeur x_{k+1} *de* x_k *comme suit*

$$x_{k+1} = x_k - \lambda_k\nabla f(x_k)$$

avec

$$\lambda_k = \frac{\bar{\lambda}}{2^t} .$$

Poser $k = k + 1$ *et aller à Etape principale* .

Fin

Soit $\{x_k\}$ *une suite générée par cet algorithme . Alors ou bien il s'arrête après un nombre fini d'itérations en un point* x_{k_0} *tel que* $\nabla f(x_{k_0}) = 0$, *ou bien il génère une suite infinie* $\{x_k\}_{k\in\mathbb{N}}$ *telle que* :

$$\nabla f(x_k) \underset{k\to\infty}{\to} 0 .$$

Chapitre 4

ALGORITHMES D'ACCELERATION DE LA CONVERGENCE DE LA METHODE DU GRADIENT

Comme on a vu dans le chapitre précédent, la méthode de la plus forte pente ou simplement la méthode du gradient travaille de façon performante dans les premières étape de l'algorithme, mais elle devient lente au voisinage d'un point statinnaire .
Pour éviter cet inconvénient nous essayerons d'accélérer la convergence de la méthode du gradient avec trois algorithmes qui sont basés sur l'epsilon algorithme, on donnera à chaque fois un résultat de convergence et on comparera grâce à des tests numériques entre ces trois algorithmes .

4.1 L'ALGORITHME EXACT EPSILON STEEPEST DESCENT

Pour construire cet algorithme, on utilise l'ε-algorithme scalaire d'ordre deux ,c'est à dire $\varepsilon_2^{(n)}$.
Etant donnée une suite $\{s_n\}$,Wynn ([36]) a montré, que les quantités $\varepsilon_2^{(n)}$ peuvent être calculées de la façon suivant :

$$\begin{aligned} \varepsilon_2^{(n)} &= \frac{s_n s_{n+2} - (s_{n+1})^2}{s_{n+2} - 2s_{n+1} + s_n}, \\ n &= 0, 1, ... \end{aligned}$$

On considère donc ici la deuxième colonne paire du tableau de l'$\varepsilon-$ algorithme scalaire, associée à la suite $\{s_n\}$.
Pour calculer $\varepsilon_2^{(n)}$ il suffit d'avoir les éléments : s_n,s_{n+1} et s_{n+2} de $\{s_n\}$.

L'algorithme exact epsilon 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 :

$$x_0 = \left(x_0^1, x_0^2, ...x_0^i, ..., x_0^n\right),$$

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

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

$$x_k = \left(x_k^1, x_k^2, ...x_k^i, ..., x_k^n\right),$$

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

$$\begin{aligned} r_k &= x_k, \\ r_k &= \left(r_k^1, r_k^2, ...r_k^i, ..., r_k^n\right). \end{aligned}$$

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

$$\begin{aligned} s_k &= \left(s_k^1, s_k^2, ...s_k^i, ..., s_k^n\right), \\ t_k &= \left(t_k^1, t_k^2, ...t_k^i, ..., t_k^n\right), \end{aligned}$$

$$s_k = r_k - \lambda_k \nabla f(r_k),$$

λ_k solution optimale de la recherche linéaire exacte

$$\underset{\lambda \geq 0}{\text{Minimiser}}\ f(r_k - \lambda_k \nabla f(r_k))$$

$$t_k = s_k - \beta_k \nabla f(s_k)$$

β_k solution optimale de la recherche linéaire exacte

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

calculer

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

Si

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

calculer

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

avec

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

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

$$x_k = \varepsilon_2^k$$

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

$$x_k = t_k$$

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

4.1.1 Convergence de l'algorithme Exacte epsilon steepest descent

Théorème 4.1 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ telle que $f \in C^1\left(\mathbb{R}^n\right)$ et (P) le problème d'optimisation sans contraintes suivant :*

$$(P) \quad \min\left\{f\left(x\right) : x \in \mathbb{R}^n\right\}$$

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

Preuve. 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

$$\Omega = \left\{x \in \mathbb{R}^n : \nabla f\left(x\right) = 0\right\}$$

On suppose le contraire, c'est à dire que $x^* \notin \Omega$.On peut définir la suite$\{x_n\}_{n\in\mathbb{N}}$ comme suit, x_0 étant un point initiale, si x_n est connu à l'itération n,x_{n+1} sera défini comme suit :

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

A est une fonction multivoque définie de cette manière

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

avec

$$\begin{aligned} B_1 &: \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n \\ x &\to B_1(x) = (x, x - \lambda_x \nabla f(x)) ; \\ \lambda_x \text{ vérifiant} &: f(x - \lambda_x \nabla f(x)) \leq f(x - \lambda \nabla f(x)) : \lambda \geq 0, \\ B_2 &: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \\ (x, y) &\to B_2(x, y) = (x, y, y - \lambda_y \nabla f(y)) ; \\ \lambda_y \text{ vérifiant} &: f(y - \lambda_y \nabla f(y)) \leq f(y - \lambda \nabla f(y)) : \lambda \geq 0 \end{aligned}$$

et

$$\begin{aligned} C &: \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n \\ (x, y, z) &\to C(x, y, z) = \omega = \left(\omega^1, ..., \omega^n\right), \end{aligned}$$

avec

$$\begin{aligned} \omega^i &= \begin{cases} \frac{x^i z^i - \left(y^i\right)^2}{z^i - 2y^i + x^i}, & \text{si } z^i - 2y^i + x^i \neq 0, \\ z^i & \text{sinon,} \end{cases} \\ x &= \left(x^1, ..., x^n\right), \ y = \left(y^1, ..., y^n\right), \ z = \left(z^1, ..., z^n\right). \end{aligned}$$

L'application B est fermée en tout point $x \notin \Omega$ et f étant continue, alors :

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

On montre que

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

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

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

Pour $n = n_0$

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

Maintenant on prend $n > n_0, n \in \mathbb{N}$ et on voit que la suite $\{f(x_n)\}_{n \in \mathbb{N}}$ est strictement décroissante, alors :

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

et donc

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

alors $\forall \varepsilon > 0, \exists n_0 \in \mathbb{N}_1, \forall n > n_0, n \in \mathbb{N}$:

$$f(x_n) - f(x^*) < \varepsilon \Rightarrow \lim_{n\to\infty, n\in\mathbb{N}} f(x_n) = f(x^*) \tag{4.1}$$

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

$$x_{n+1} = C(x_n, y_n, z_n), \quad (x_n, y_n, z_n) \in B(x_n), \tag{4.2}$$

dans l'algorithme $x_n = r_n, y_n = s_n$ et $z_n = t_n$.
On remarque que les suites :$\{x_n\}, \{y_n\}, \{z_n\}$ et $\{x_{n+1}\}$ appartiennent à l'ensemble compact $\delta(x_0)$,alors il existe $\mathbb{N}_2 \subset \mathbb{N}_1$ tels que

$$x_n \underset{n\to\infty, n\in\mathbb{N}_2}{\to} x^* \tag{4.3}$$

$$y_n \underset{n\to\infty, n\in\mathbb{N}_2}{\to} y^* \tag{4.4}$$

$$z_n \underset{n\to\infty, n\in\mathbb{N}_2}{\to} z^* \tag{4.5}$$

$$x_{n+1} \underset{n\to\infty, n\in\mathbb{N}_2}{\to} \hat{x}. \tag{4.6}$$

Rappelons que B est fermée au point x^* .Les relations (4.3),.(44),(4.5),et (4.6) donnent :

$$(x^*, y^*, z^*) \in B(x^*). \tag{4.7}$$

$x^* \notin \Omega\ (\nabla f(x^*) \neq 0)$ alors $-\nabla f(x^*)$ est une direction de descente. D'après les définitions des fonctions multivoques B, B_1 et B_2

$$f(y^*) < f(x^*) \tag{4.8}$$

$$f(z^*) < f(x^*). \tag{4.9}$$

étant donné

$$x_{n+1} = C(x_n, y_n, z_n), \tag{4.10}$$

alors selon l'algorithme

$$f(x_{n+1}) \leq f(x_n) \tag{4.11}$$

$$f(x_{n+1}) \leq f(y_n) \tag{4.12}$$

$$f(x_{n+1}) \leq f(z_n). \tag{4.13}$$

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

$$\begin{aligned} f(\hat{x}) &\leq f(x^*) \\ f(\hat{x}) &\leq f(y^*) \\ f(\hat{x}) &\leq f(z^*). \end{aligned} \tag{4.14}$$

Les relations $(4.8), (4.9)$ impliquent que

$$f(\hat{x}) < f(x^*). \tag{4.15}$$

D'autre part on a

$$\lim_{n\to\infty, n\in\mathbb{N}_2} f(x_{n+1}) = f(\hat{x}) \tag{4.16}$$

et

$$\lim_{n\to\infty, n\in\mathbb{N}} f(x_n) = f(x^*) \text{ ,voir } (4.1)$$

alors on obtient

$$f(x^*) = f(\hat{x}), \tag{4.17}$$

ce qui est en contradiction avec (4.15), donc $x^* \in \Omega$ c'est à dire $\nabla f(x^*) = 0$. ■

4.2 L'ALGORITHME ARMIJO EPSILON STEEPEST DESCENT

Pour construire cet algorithme, on utilise la colonne $\varepsilon_2^{k,i}$ $(i = 1, 2, .., n)$.
Etant donnée la suite $\{x_k^i\}_{k\in\mathbb{N}}$ $(i = 1, 2, .., n)$. Les quantités $\varepsilon_2^{k,i}$ peuvent être calculées de la façon suivante :

$$\varepsilon_2^{k,i} = x_{k+1}^i + \left[\frac{1}{x_{k+2}^i - x_{k+1}^i} - \frac{1}{x_{k+1}^i - x_k^i}\right]^{-1}.$$

Cette formule était proposée par F.Cordellier [12],elle est numériquement plus efficace que la version de P.Wynn([36], [37]).

L'algorithme Armijo epsilon steepest descent :

Initialisation : On choisit un point initial $x_0 \in \mathbb{R}^n$.Avec

$$x_0 = \left(x_0^1, x_0^2, ..., x_0^i, ..., x_0^n\right) \in \mathbb{R}^n.$$

Poser $k = 0$ et aller à l'étape principale.

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

$$x_k = \left(x_k^1, x_k^2, ..., x_k^i, ..., x_k^n\right).$$

Si $\|\nabla f(x_k)\| = 0$ stop, sinon poser $r_k = x_k$ et calculons s_k et t_k en utilisant deux fois la méthode du gradient avec la recherche linéaire inexacte d'Armijo

$$s_k = r_k - \lambda_k \nabla f(r_k)$$

et

$$t_k = s_k - \beta_k \nabla f(s_k)$$

λ_k et β_k sont des solutions optimals obtenus avec la recherche linéaire d'Armijo.
Calculer

$$s_k^i - r_k^i; t_k^i - s_k^i; \frac{1}{t_k^i - s_k^i} - \frac{1}{s_k^i - r_k^i}, \quad i = 1, 2, .., n.$$

Si

$$s_k^i - r_k^i \neq 0 \,; t_k^i - s_k^i \neq 0 \,; \frac{1}{t_k^i - s_k^i} - \frac{1}{s_k^i - r_k^i} \neq 0 \,, \quad i = 1, 2, .., n$$

alors calculer

$$\varepsilon_2^{k,i} = s_k^i + \left[\frac{1}{t_k^i - s_k^i} - \frac{1}{s_k^i - r_k^i}\right]^{-1}, \quad i = 1, 2, .., n$$

et

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

Si $f\left(\varepsilon_2^k\right) < f(t_k)$, poser $x_k = \varepsilon_2^k$, remplacer k par $k+1$ et aller à l'étape principale.
Si $f\left(\varepsilon_2^k\right) \geq f(t_k)$ ou bien si $s_k^{i_0} - r_k^{i_0} = 0$, ou $t_k^{i_0} - s_k^{i_0} = 0$ ou $\frac{1}{t_k^{i_0} - s_k^{i_0}} - \frac{1}{s_k^{i_0} - r_k^{i_0}} = 0$, $i_0 \in \{1, 2, .., n\}$.
Poser $x_k = t_k$, remplacer k par $k+1$ et aller à l'étape principale.

Remarque 4.1 *Si $f\left(\varepsilon_2^k\right) < f(t_k)$ alors : $f(x_{k+1}) = f\left(\varepsilon_2^k\right) < f(x_k)$ et si $f\left(\varepsilon_2^k\right) \geq f(t_k)$ ou bien le calcule de ε_2^k n'est pas possible alors d'apres l'algorithme on a :*

$$f(x_{k+1}) = f(t_k) < f(x_k)$$

c'est à dire :l' Armijo epsilon steepest descente algorithme garanti :

$$f(x_{k+1}) < f(x_k), \qquad k = 0, 1, 2, ...$$

4.2.1 Convergence globale de l'algorithme Armijo epsilon steepest descent

Théorème 4.2 *Soit* $f : \mathbb{R}^n \to \mathbb{R}$ *continument différentiable dans l'ensemble*

$$\delta(x_0) = \{x \in \mathbb{R}^n : f(x) < f(x_0), x_0 \in \mathbb{R}^n\}.$$

$\nabla f(x)$ *est lipschitzien dans* $\delta(x_0)$ $(i.e)$

$$\exists M > 0 : \|\nabla f(x) - \nabla f(y)\| \leq M\|x - y\| \quad \forall (x, y) \in \delta(x_0)$$

soit $\{x_k\}$ *une suite générée par Armijo* $\varepsilon-$*steepest algorithme, alors cet algorithme s'arrête après un nombre fini d'itérations en un point* x_{k_0} *tel que* $\nabla f(x_{k_0}) = 0$ *ou bien il génère une suite infinie* $\{x_k\}_{k\in\mathbb{N}}$ *telle que*

$$\nabla f(x_k) \underset{k\to\infty}{\to} 0$$

Preuve. Supposons que l'algorithme Armijo epsilon steepest descent génère une suite infinie $\{x_k\}_{k\in\mathbb{N}}$.
A l'étape principale de l'algorithme on applique deux fois la méthode du gadient avec la recherche linéaire d'Armijo.
s_k et t_k sont les successeurs de x_k qui permettant par la suite de trouver x_{k+1}.

$$s_k = x_k - \lambda_k \nabla f(x_k)$$

$\lambda_k = \frac{\bar{\alpha}}{2^n}$ ($\bar{\alpha} > 0$ constante définie dans la recherche linéaire inexact d'Armijo) vérifie :

$$\varphi\left(\frac{\bar{\alpha}}{2^n}\right) \leq \hat{\varphi}\left(\frac{\bar{\alpha}}{2^n}\right) \tag{4.18}$$

avec

$$\varphi\left(\frac{\bar{\alpha}}{2^n}\right) = f\left(x_k - \frac{\bar{\alpha}}{2^n}\nabla f(x_k)\right) \tag{4.19}$$

et

$$\hat{\varphi}\left(\frac{\bar{\alpha}}{2^n}\right) = f(x_k) - \frac{\bar{\alpha}}{2^n} c \nabla f(x_k)^t \nabla f(x_k) \tag{4.20}$$

$c \in]0, 1[, \bar{\alpha} > 0$ et $n \in \mathbb{N}$, tel que (4.18) est satisfaite.
Aors d'après $(4.18), (4.19)$ et (4.20) on obtient

$$\begin{aligned} f(s_k) &= f\left(x_k - \frac{\bar{\alpha}}{2^n}\nabla f(x_k)\right) \\ &\leq f(x_k) - \frac{\bar{\alpha}}{2^n} c \|\nabla f(x_k)\|^2 \end{aligned} \tag{4.21}$$

(4.21) impliques

$$f(s_k) - f(x_k) \leq -\frac{\bar{\alpha}}{2^n} c \|\nabla f(x_k)\|^2 \tag{4.22}$$

D'autre part en dévloppant f au voisinage du point x_k, on obtiet :

$$f(s_k) - f(x_k) = (s_k - x_k)^t \nabla f(\tilde{x}); \quad \tilde{x} = \lambda s_k + (1-\lambda) x_k, \quad \lambda \in]0,1[\tag{4.23}$$

ou encore remarquant que

$$s_k = x_k - \alpha_k \nabla f(x_k), \ \alpha_k = \frac{\bar{\alpha}}{2^n} \tag{4.24}$$

alors

$$\begin{aligned} f(s_k) - f(x_k) &= -\alpha_k \nabla f(x_k)^t . \nabla f(\tilde{x}) \\ &= -\alpha_k \nabla f(x_k)^t \left[\nabla f(x_k) - \nabla f(x_k) + \nabla f(\tilde{x})\right] \\ &= -\alpha_k \left\|\nabla f(x_k)\right\|^2 + \alpha_k \nabla f(x_k)^t \left[\nabla f(x_k) - \nabla f(\tilde{x})\right] \end{aligned} \tag{4.25}$$

$\tilde{x} = \lambda s_k + (1-\lambda) x_k$, $\lambda \in]0,1[$.
Notons que la suite $\{f(x_k)\}_{k\in\mathbb{N}}$ est décroissante, alors

$$x_k \in \delta(x_0), \quad k = 0, 1, ...$$

En utilisant l'inégalité de Cauchy Schwarz et le fait que le gradient de f est Lipshitzien de constante M, on obtient

$$\begin{aligned} f(s_k) - f(x_k) &\leq -\alpha_k \left\|\nabla f(x_k)\right\|^2 + \alpha_k \left\|\nabla f(x_k)\right\| \left\|\nabla f(x_k) - \nabla f(\tilde{x})\right\| \\ &\leq -\alpha_k \left\|\nabla f(x_k)\right\|^2 + \alpha_k \left\|\nabla f(x_k)\right\| M \left\|x_k - \tilde{x}\right\| \end{aligned} \tag{4.26}$$

Notons que $\tilde{x} = \lambda s_k + (1-\lambda) x_k$, $\lambda \in]0,1[$, alors

$$\left\|x_k - \tilde{x}\right\| = \left\|(1-\lambda)(x_k - x_s)\right\| \leq \left|(1-\lambda)\right| \left\|(x_k - x_s)\right\| < \left\|(x_k - x_s)\right\|. \tag{4.27}$$

(4.26) et (4.27) impliquent

$$f(s_k) - f(x_k) \leq -\alpha_k \left\|\nabla f(x_k)\right\|^2 + \alpha_k \left\|\nabla f(x_k)\right\| M \left\|(x_s - x_k)\right\|$$

On remarque que

$$x_s - x_k = x_k - \alpha_k \nabla f(x_k) - x_k = -\alpha_k \nabla f(x_k), \ \alpha_k = \frac{\bar{\alpha}}{2^n},$$

alors

$$f(s_k) - f(x_k) \leq -\alpha_k \left\|\nabla f(x_k)\right\|^2 + \alpha_k \left\|\nabla f(x_k)\right\| M \alpha_k \left\|\nabla f(x_k)\right\|. \tag{4.28}$$

On obtient

$$f(s_k) - f(x_k) \leq -\frac{\bar{\alpha}}{2^n} \left\|\nabla f(x_k)\right\|^2 \left[1 - M\frac{\bar{\alpha}}{2^n}\right]. \tag{4.29}$$

On choisit maintenant le petit entier $n > 0$ telque :

$$2^n \geq \frac{M\bar{\alpha}}{1-c} \tag{4.30}$$

(4.30) implique

$$1 - M\frac{\bar{\alpha}}{2^n} \geq c \tag{4.31}$$

Alors

$$1 - M\frac{\bar{\alpha}}{2^{n-1}} < c \tag{4.32}$$

(4.31) implique

$$-\frac{\bar{\alpha}}{2^n}\left\|\nabla f(x_k)\right\|^2\left[1 - M\frac{\bar{\alpha}}{2^n}\right] \leq -\frac{\bar{\alpha}}{2^n}c\left\|\nabla f(x_k)\right\|^2, c \in]0,1[\tag{4.33}$$

et (4.32) nous donne

$$-M\frac{\bar{\alpha}}{2^{n-1}} < c - 1 \Rightarrow M\frac{\bar{\alpha}}{2^{n-1}} > 1 - c$$

Alors

$$-\frac{\bar{\alpha}}{2^n}c < -\frac{c(1-c)}{2M} \tag{4.34}$$

Si le choix de $n > 0$ vérifie(4.30), l'inégalité (4.22) reste vraie. Prenons en considération les relations $(4.29), (4.31), (4.32), (4.33), (4.34)$ on obtient :

$$f\left(s_k\right) - f(x_k) \leq -\frac{c(1-c)}{2M}\left\|\nabla f(x_k)\right\|^2 \tag{4.35}$$

Posons

$$G = -\frac{c(1-c)}{2M}$$

il est clair que $G < 0$, alors(4.35) donne :

$$f\left(s_k\right) - f(x_k) \leq G\left\|\nabla f(x_k)\right\|^2 \tag{4.36}$$

De la même manière pour la deuxième application de la méthode du gradient avec la recherche linéaire inexacte d'Armijo on obtient :

$$f\left(t_k\right) - f(s_k) \leq G\left\|\nabla f(s_k)\right\|^2 \tag{4.37}$$

Il reste de montrer que :

$$\lim_{k\to\infty}\left\|\nabla f(x_k)\right\| = 0$$

Pour cela, considérons :

$$f\left(x_{k+1}\right) - f(x_k) = f(\varepsilon_k) - f(t_k) + f(t_k) - f(s_k) + f(s_k) - f(x_k) < f(t_k) - f(s_k) + f(s_k) - f(x_k)$$

car $f(\varepsilon_k) - f(t_k) < 0$

D'après les relations (4.36) et (4.37) on

$$f\left(x_{k+1}\right)-f(x_k)<G\left(\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2\right) \tag{4.38}$$

la suite $\{f(x_k)\}_{k\in\mathbb{N}}$ est strictement décroissante, minoré,donc convergente,la relation (4.38) implique :

$$\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2<\frac{1}{G}\left[f\left(x_{k+1}\right)-f(x_k)\right] \tag{4.39}$$

Par passage à $\overline{\lim\limits_{k\to\infty}}$ on obtient :

$$\overline{\lim_{k\to\infty}}\left(\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2\right)\leq 0 \tag{4.40}$$

D'autre part on a

$$\underline{\lim_{k\to\infty}}\left(\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2\right)\leq\overline{\lim_{k\to\infty}}\left(\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2\right) \tag{4.41}$$

et

$$0\leq\underline{\lim_{k\to\infty}}\left(\|\nabla f(s_k)\|^2+\|\nabla f(x_k)\|^2\right). \tag{4.42}$$

Les inegalités (4.40) , (4.41) , et (4.42) impliquent :

$$\begin{aligned}0 &\leq \underline{\lim_{k\to\infty}}\left(\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2\right)\\ &\leq \overline{\lim_{k\to\infty}}\left(\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2\right)\leq 0\end{aligned} \tag{4.43}$$

ceci implique :

$$\begin{aligned}&\underline{\lim_{k\to\infty}}\left(\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2\right)=\overline{\lim_{k\to\infty}}\left(\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2\right)\\ =\ &\lim_{k\to\infty}\left(\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2\right)=0.\end{aligned}$$

et on a

$$0\leq\|\nabla f(x_k)\|^2\leq\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2$$

et

$$0\leq\|\nabla f(s_k)\|^2\leq\|\nabla f(x_k)\|^2+\|\nabla f(s_k)\|^2.$$

Finallement on obtient

$$\lim_{k\to\infty}\|\nabla f(x_k)\|=\lim_{k\to\infty}\|\nabla f(s_k)\|=0.$$

■

Dans [14] Degaichia et Benzine ont accéléré la convergence de la méthode du gradient et ont démontré la convergence globale en utilisant l'epsilon algorithme et la recherche linéaire inexacte de Wolfe . Ils ont appelé le nouveau algorithme : Wolfe epsilon steepest descent algorithm .

4.3 L'ALGORITHME WOLFE EPSILON STEEPEST DESCENT

On utilise les colonnes $\varepsilon_2^{k,i}$ $(i=1,2,...,n)$ pour construire cet algorithme. Etant donnée la suite $\{x_k^i\}_{k\in\mathbb{N}}$ $(i=1,2,...,n)$,les $\varepsilon_2^{k,i}$ $(i=1,2,...,n)$ sont calculées avec la formule de F.Cordelier [12] :

$$\varepsilon_2^{k,i}=x_{k+1}^i+\left[\frac{1}{x_{k+2}^i-x_{k+1}^i}-\frac{1}{x_{k+1}^i-x_k^i}\right]^{-1},(i=1,2,...,n)$$

Pour calculer $\varepsilon_2^{k,i}$ on utilise x_k^i, x_{k+1}^i et x_{k+2}^i $(i=1,2,...,n)$.

L'algorithme Wolfe epsilon steepest descent :

Initialisation : Choisir un vecteur initial $x_0\in\mathbb{R}^n$. Avec

$$x_0=\left(x_0^1,x_0^2,...,x_0^i,...,x_0^n\right)\in\mathbb{R}^n.$$

Poser $k=0$ et aller à l'étape principale.

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

$$x_k=\left(x_k^1,x_k^2,...,x_k^i,...,x_k^n\right).$$

Si $\|\nabla f(x_k)\|=0$ stop, sinon poser $r_k=x_k$ et calculons s_k et t_k en utilisant deux fois la méthode du gradient avec la recherche linéaire inexacte de Wolfe $(i.e)$

$$s_k=r_k-\lambda_k\nabla f(r_k)$$

et

$$t_k=s_k-\beta_k\nabla f(s_k)$$

λ_k et β_k sont des scalaires obtenus avec la recherche linéaire inexacte de Wolfe.

Calculer

$$s_k^i-r_k^i;t_k^i-s_k^i;\frac{1}{t_k^i-s_k^i}-\frac{1}{s_k^i-r_k^i},\quad i=1,2,..,n.$$

Si

$$s_k^i-r_k^i\neq 0\,;t_k^i-s_k^i\neq 0\,;\frac{1}{t_k^i-s_k^i}-\frac{1}{s_k^i-r_k^i}\neq 0\,,\quad i=1,2,..,n$$

Posons

$$\varepsilon_2^{k,i}=s_k^i+\left[\frac{1}{t_k^i-s_k^i}-\frac{1}{s_k^i-r_k^i}\right]^{-1},\quad i=1,2,..,n$$

et

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

Si $f\left(\varepsilon_2^k\right)<f(t_k)$, poser $x_k=\varepsilon_2^k$, remplacer k par $k+1$ et aller à l'étape principale.

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

4.3.1 Convergence globale de l'algorithme wolfe epsilon steepest descent

Théorème 4.3 *Considérons le problème de minimisation sans contrainte* (P) *et* x_1 *le point de départ de l'Algorithme Wolfe epsilon steepest descente. Supossons aussi que les conditions suivante soient satisfaites :*
f *est minorée dans* $\mathbb{R}^n$ *et que* f *est continument différentiable dans un voisinage* N *de l'ensemble* $L = \{x : f(x) \leq f(x_1)\}$.
Supposons aussi que le gradient est Lipshitzien $(i.e)$ *il existe* $G > 0$ *telle que*

$$\| g(x) - g(y) \| \leq G \| x - y \| \quad \text{pour } x, y \in \mathbb{N}.$$

Alors la suite $\{x_k\}_{k\in\mathbb{N}}$ *générée par l'algorithme Wolfe epsilon steepest descente satisfait l'une des deux conditions suivantes :*

$$\nabla f(x_{k_0}) = 0$$

pour un certain indice $k_0 \in \mathbb{N}$ *ou*

$$\| \nabla f(x_k) \| \underset{k\to\infty}{\to} 0.$$

Preuve. *On suppose que l'algorithme Wolfe steepest descent génère une suite infinie* $\{x_k\}_{k\in\mathbb{N}}$.
D'après l'algorithme ,les vecteurs s_k *et* t_k *sont obtenus en utilisant deux fois la méthode du gradient avec la recherche linéaire inexacte de Wolfe. Alors on a*

$$f(s_k) < f(r_k) = f(x_k)$$

et

$$f(t_k) < f(s_k)$$

Si le calcul de ε_2^k *est possible alors on a deux cas :*

a) $f(\varepsilon_2^k) < f(t_k)$ *.Alors on a*

$$f(x_{k+1}) = f(\varepsilon_2^k) < f(x_k).$$

ou bien

b) $f(\varepsilon_2^k) \geq f(t_k)$ *ou le calcule de* ε_2^k *n'est pas possible.*

Dans ce cas on a

$$f(x_{k+1}) = f(t_k) < f(x_k).$$

En tous cas :

$$f(x_{k+1}) < f(x_k),\ k = 0, 1, 2, ... \tag{4.44}$$

D'autre part on a

$$\varepsilon_2^k = x_{k+1} + \alpha_k d_k \tag{4.45}$$

où d_k *et* α_k *dépend de* s_k,r_k,t_k,λ_k *et* β_k.

(4.44) *implique que* d_k *est une direction de descente et donc d'après le théorème de Zoudentijk on a*

$$\sum_{k=0}^{\infty} \cos^2(\theta_k) \parallel g_k \parallel^2 < \infty \tag{4.46}$$

(4.46)*implique*

$$\lim_{k\to\infty} \cos^2(\theta_k) \parallel g_k \parallel^2 = 0 \tag{4.47}$$

Il n'est pas difficile de démontrer qu'il existe $\delta > 0$ *telle que :*

$$\cos(\theta_k) \geq \delta > 0 \tag{4.48}$$

pour tout k.

Notons

$$u_k = \cos^2(\theta_k), v_k = \|g_k\|^2, w_k = u_k.v_k \tag{4.49}$$

$(4.48), (4.49)$ *impliquent*

$$u_k \neq 0, v_k = \frac{w_k}{u_k} \tag{4.50}$$

et (4.47) *nous donne*

$$\lim_{k\to\infty} w_k = 0 \tag{4.51}$$

(4.48) *et* (4.50) *impliquent*

$$0 \leq v_k \leq \frac{1}{\delta^2} w_k \tag{4.52}$$

et $(4.51), (4.52)$ *donnent*

$$\lim_{k\to\infty} v_k = 0$$

et donc

$$\lim_{k\to\infty} \|g_k\| = 0$$

■

4.4 RESULTATS NUMERIQUES ET COMPARAISONS

4.4.1 Résultat 1 :[18]

Les tests numériques effectués permettent de comparer l'algorithme de la plus forte pente avec les ε-steepest descent algorithmes .Les fonctions tests utilisées pour cette comparaison sont : la

fonction de Rosenbrock et la fonction de Oren . Les tests numériques que nous présentons ici sont très significatifs en ce qui concerne l'accélération de la convergence .

Problème1 : (la fonction Rosenbrock)

$$f(x)=\sum_{i=1}^{n-1}\left\{100\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i\right)^2\right\}. \text{ avec } x_0=(-1,1,...,(-1)^n),\ x^*=(1,...,1),\ f(x^*)=0.$$

Problème2 : (la fonction de Oren)

$$f(x)=\left[\sum_{i=1}^{n} ix_i^2\right]^2,\ x_0=(1,1,...,1),\ x^*=(0,...,0),\ f(x^*)=0.$$

Problème3 : (la fonction Pen1)

$$f(x)=\sum_{i=1}^{n}\left(x_i-1\right)^2+10^{-3}\left(\sum_{i=1}^{n}x_i^2-0,25\right)^2,\quad x_0=\left(\frac{1}{n+1},...,\frac{n}{n+1}\right),\ x^*=(1,-1,1,-1,...)^t.$$

Problème4 :(la fonction trigonométrique)

$$\begin{aligned} f(x) &= \sum_{i=1}^{n}\left[n+i\left(1-\cos x_i\right)-\sin x_i-\sum_{j=1}^{n}\cos x_j\right]^2 \\ x_0 &= \left(\frac{1}{5n},...,\frac{1}{5n}\right),\ x^*=(0,0,0,...,0),\ f(x^*)=0. \end{aligned}$$

Dans cette partie, on va adopter les notations suivantes:

n : la taille du problème.

N : nombre des itération.

F_* : la valeur de la fonction objective.

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

Comparaison de la méthode de la plus forte pente avec l'$\varepsilon-$steepest descent algorithme (**Wynn et Gordellier**)

Résultats numériques pour la fonction de Rosenbrock.

Tableau 4.1: **steepest descent avec recherche linéaire exacte** (**dichotomie**) .

k	t_k	x_k	$f(x_k)$	$\| g(x_k) \|$
1	$-$	$(-1.20, 1.000)$	24.200	232.867
10	4.777×10^{-03}	$(-1.009, 1.019)$	4.037	2.928
100	3.811×10^{-03}	$(-0.771, 0.596)$	3.138	3.092
200	8.999×10^{-03}	$(0.147, 0.018)$	0.727	1.642
500	1.634×10^{-03}	$(0.807, 0.650)$	3.726×10^{-02}	0.220
1000	1.243×10^{-03}	$(0.944, 0.891)$	3.095×10^{-03}	5.547×10^{-02}
1500	1.161×10^{-03}	$(0.981, 0.962)$	3.590×10^{-04}	1.826×10^{-02}
2000	1.136×10^{-03}	$(0.993, 0.986)$	4.538×10^{-05}	6.4229×10^{-03}

Tableau 4.2: steepest descent avec recherche linéaire d'Armijo.

k	t_k	x_k	$f(x_k)$	$\| g(x_k) \|$
1	$-$	$(-1.200, 1.000)$	24.200	232.867
10	2.499×10^{-03}	$(-1.016, 1.035)$	4.067	3.239
100	4.999×10^{-03}	$(0.360, 0.123)$	0.413	1.399
200	2.499×10^{-03}	$(0.683, 0.466)$	0.100	0.409
500	2.499×10^{-03}	$(0.930, 0.865)$	4.841×10^{-03}	0.124
1000	1.250×10^{-03}	$(0.976, 0.953)$	5.501×10^{-04}	2.304×10^{-02}
1500	2.499×10^{-03}	$(0.988, 0.977)$	1.225×10^{-04}	1.390×10^{-02}
2000	4.999×10^{-03}	$(0.994, 0.989)$	2.838×10^{-05}	1.037×10^{-02}

Tableau 4.3: steepest descent avec recherche linéaire de Goldestein.

k	t_k	x_k	$f(x_k)$	$\| g(x_k) \|$
1	$-$	$(-1.200, 1.000)$	24.200	232.867
10	9.024×10^{-03}	$(-0.939, 0.878)$	3.765	5.702
100	8.573×10^{-04}	$(-0.483, 0.248)$	2.221	2.956
200	8.573×10^{-04}	$(0.386, 0.149)$	0.376	1.221
500	8.573×10^{-04}	$(0.737, 0.543)$	6.894×10^{-02}	0.428
1000	9.024×10^{-03}	$(0.906, 0.821)$	8.749×10^{-03}	0.142
1500	8.573×10^{-04}	$(0.989, 0.978)$	1.117×10^{-04}	1.356×10^{-02}
2000	8.573×10^{-04}	$(0.998, 0.996)$	2.716×10^{-06}	1.478×10^{-03}

Tableau 4.4: steepest descent avec recherche linéaire de Wolfe.

k	t_k	x_k	$f(x_k)$	$\| g(x_k) \|$
1	$-$	$(-1.200, 1.000)$	24.200	232.867
10	2.499×10^{-03}	$(-1.016, 1.03)$	4.067	3.239
100	4.999×10^{-03}	$(0.593, 0.347)$	0.166	0.762
200	2.499×10^{-03}	$(0.716, 0.511)$	8.059×10^{-02}	0.455
500	2.000×10^{-02}	$(0.927, 0.859)$	5.380×10^{-03}	0.332
1000	9.999×10^{-03}	$(0.975, 0.951)$	5.919×10^{-04}	8.178×10^{-02}
1500	9.999×10^{-03}	$(0.988, 0.977)$	1.279×10^{-04}	1.259×10^{-02}
2000	1.250×10^{-03}	$(0.994, 0.989)$	2.887×10^{-05}	6.108×10^{-03}

Tableau 4.5: $\varepsilon-$**steepest descent algorithme** (**Wynn**) **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	$1,0492 \times 10^{-5}$	$8,7060 \times 10^{-11}$
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 4.6: $\varepsilon-$**steepest descent algorithme** (**Wynn**) **avec la recherche linéaire deGoldestein.**

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	$1,8792 \times 10^{-5}$	$2,0050 \times 10^{-11}$
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	$7,2350 \times 10^{-5}$	$3,0391 \times 10^{-11}$

Tableau 4.7: $\varepsilon-$**steepest descent algorithme** (**Wynn**) **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	$1,1304 \times 10^{-5}$	$1,1384 \times 10^{-10}$
10000	151	$1,1027 \times 10^{-5}$	$3,0391 \times 10^{-10}$

Tableau 4.8 : $\varepsilon-$**steepest descent algorithme** (**Cordellier**) **avec la recherche linéaire d'Armijo.**

n	N	$\|\nabla F\|$	F_*
2	41	$0,6 \times 10^{-5}$	$0,46 \times 10^{-10}$
10	43	$0,81 \times 10^{-5}$	$0,83 \times 10^{-10}$
100	47	$0,92 \times 10^{-5}$	$1,07 \times 10^{-10}$
1000	53	$0,75 \times 10^{-5}$	$0,71 \times 10^{-10}$
2000	55	$0,64 \times 10^{-5}$	$0,51 \times 10^{-10}$

Résultats numériques pour la fonction de Oren.

Tableau 4.9 : Steepest descent avec la recherche linéaire d'Armijo.

n	N	$\|\nabla F\|$	F_*
2	6	$0,90 \times 10^{-7}$	$0,61 \times 10^{-10}$
10	14	$0,14 \times 10^{-5}$	$0,21 \times 10^{-8}$
100	67	$0,69 \times 10^{-5}$	$0,16 \times 10^{-7}$
1000	477	$0,41 \times 10^{-5}$	$0,68 \times 10^{-9}$
2000	1061	$0,18 \times 10^{-5}$	$0,22 \times 10^{-10}$
10000	$N \geq 5000$	/	/

Tableau 4.10: $\varepsilon-$steepest descent algorithme (Wynn) avec la recherche linéaire d'Armijo.

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

Tableau 4.11: $\varepsilon-$steepest descent algorithme (Cordellier) avec la recherche linéaire d'Armijo.

n	N	$\|\nabla F\|$	F_*
2	7	$0,51 \times 10^{-5}$	$0,13 \times 10^{-7}$
10	10	$0,42 \times 10^{-5}$	$0,10 \times 10^{-7}$
100	25	$0,72 \times 10^{-5}$	$0,22 \times 10^{-7}$
1000	53	$0,98 \times 10^{-5}$	$0,33 \times 10^{-7}$
2000	89	$0,73 \times 10^{-5}$	$0,22 \times 10^{-7}$

Tableau 4.12: $\varepsilon-$**steepest descent algorithme** (**Cordellier**) **avec la recherche linéaire de Wolfe.**

n	N	$\|\nabla F\|$	F_*
2	7	$5,104 \times 10^{-6}$	$1,383 \times 10^{-8}$
10	10	$4,227 \times 10^{-6}$	$1,075 \times 10^{-8}$
100	25	$7,421 \times 10^{-6}$	$2,279 \times 10^{-8}$
1000	53	$9,834 \times 10^{-6}$	$3,316 \times 10^{-8}$
2000	84	$7,330 \times 10^{-6}$	$2,242 \times 10^{-8}$

Résultats numériques pour la fonction de Pen1.

Tableau 4.13: **Steepest descent avec la recherche linéaire d'Armijo.**

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

Tableau 4.14: $\varepsilon-$**steepest descent algorithme** (**Wynn**) **avec la recherche linéaire d'Armijo.**

n	N	$\|\nabla F\|$	F_*
50	6	$0,42 \times 10^{-6}$	$0,21 \times 10^{+1}$
100	7	$0,78 \times 10^{-5}$	$0,74 \times 10^{+1}$
1000	24	$0,81 \times 10^{-5}$	$0,29 \times 10^{+3}$
3000	22	$0,35 \times 10^{-5}$	$0,13 \times 10^{+4}$
10000	31	$0,97 \times 10^{-5}$	$0,57 \times 10^{+4}$

Résultats numériques pour la fonction trigonométrique.

Tableau 4.15: **Steepest descent avec la recherche linéaire d'Armijo.**

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

Tableau 4.16: $\varepsilon-$steepest descent algorithme (Wynn) avec la recherche linéaire d'Armijo.

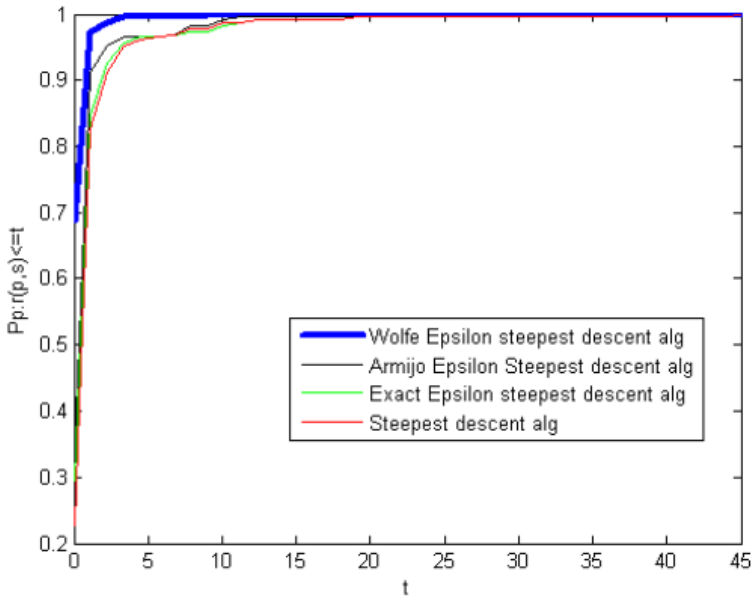
n	N	$\|\nabla F\|$	F_*
10	5	$0,14 \times 10^{-6}$	$0,49 \times 10^{-14}$
50	4	$0,27 \times 10^{-5}$	$0,18 \times 10^{-11}$
100	4	$0,17 \times 10^{-5}$	$0,69 \times 10^{-12}$
1000	4	$0,47 \times 10^{-6}$	$0,54 \times 10^{-13}$

4.4.2 Résultat 2 :[14]

On va exposer dans cette partie des résultats numériques obtenus par implémentation de la méthode Wolfe Epsilon Steepest Descent algorithm. Pour obtenir ces résultats, Degaichia et Benzine ont utilisés les fonctions tests et leurs programmes en fortran tirés de ([2]). Ils ont utilisé les mêmes critères que ceux utilisés dans ([3]). Les programmes ont été écrits en Fortran 90 et compilés dans une station Intel Pen- tium 4 with 2 GHz. ils ont choisi 52 fonctions test pour des problèmes sans contraintes. Pour chaque fonction test, nous avons associé un nombre de variables croissant $n = 2, 10, 30, 50, 70, 100, 300, 500, 700, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000$.
On obtient ainsi 962 tests numériques. L' algorithme implémente la méthode Wolfe Epsilon Steepest Descent algorithm avec les conditions de Wolfe $(1.4), (1.5)$ ([34], [35]). L'algorithme s'arrête quand $\|\nabla f(x_k)\| < 10^{-6}$.
Pour comparer des algorithmes, ils ont utilisés le critère suivant : Soient f_i^{ALG1} et f_i^{ALG2} les valeurs optimales de f obtenus respectivement par $ALG1$ et $ALG2$, pour les problèmes $i = 1, ..., 962$. Pour le test i, nous dirons que l'algorithme1 : $ALG1$ est plus performent que le deuxième algorithme2 : $ALG2$ si :

$$\left|f_i^{ALG1} - f_i^{ALG2}\right| < 10^{-3}$$

et si le temps CPU accompli par $ALG1$ est inferieur au temps CPU correspondant à $ALG2$. On pourrait comparer aussi la performance par le nombre d'itérations.
Nous comparons à travers les tests numériques effectués, l'algorithme Wolfe Epsilon Steepest Descent algorithm et l'Algorithme Armijo Epsilon Steepest descent, l'algorithme Epsilon steepest descent algorithm version recherche linéaire exacte et enfin la méthode du gradient. Dans la Figure 1 on trouve le procédé de Dolan et Moré CPU performance. Ce procédé montre clairement que la méthode Wolfe Epsilon Steepest Descent algorithm est plus performente que l'Algorithme Armijo Epsilon Steepest descent, l'algorithme Epsilon steepest descent algorithm version recherche linéaire exacte et la méthode du gradient.



$fig1$

4.4.3 Conclusion

D'après les tests numériques on a prouvé que la suite génerée par l'ε-steepest descent algorithme avec la formule de Cordellier donne de meilleurs résultats que l'ε-steepest descent algorithme avec la formule de Wynn et que Wolfe epsilon steepest descent est plus performant que les deux autres .

Bibliographie

[1] Aitken, A.C. "OnBernoulli's numerical solution of algebraic equations", Proc. Roy. Soc. Edinburgh, 46, pp.$289 - 305$, 1926.

[2] Andrei, N. An unconstrained optimization test functions collection. Advanced Modeling and Optimization, 10 (2008), pp. $147 - 161$.

[3] Andrei, N. Another conjugate gradient algorithm for unconstrained optimization. Annals of Academy of Romanian Scientists, Series on Science and Technology of Information, vol. 1, nr.1, 2008, pp.$7 - 20$.

[4] Armijo, L. " Minimization of functions having Lipschitz continuous first partial derivatives," Pacific J. Mathematics, 16, 1, pp.$1 - 3$, 1966.

[5] Bongartz, I, Conn, A.R., Gould, N.I.M., Toint, P.L., "Cute: constrained and unconstrained testing environments", ACM transactions on Mathematical software $21(1995)$ $123 - 160$.

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

[7] Bongartz, I., Conn, A.R., Gould, N.I.M. and Toint, P.L. CUTE : constrained and unconstrained testing environments, ACM Trans. Math. Software, 21, pp.$123 - 160$, 1995.

[8] Broyden, C.G. "Quasi-Newton Methods and their application to Function Minimization "Mathematics of Coputation, 21, pp. $368 - 381$, 1967.

[9] Broyden, C.G. " The convergence of a class of double rank minimization algorithms 2. The new algorithm. "J. Institute of Mathematics and its applications, 6, pp. $222 - 231$, 1970.

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

[11] Cauchy, A.L. "*Méthode générale pour la résolution des systèmes d'équations simul*$\tan$*ées*",C.R. Acad. Sci. Paris, 25, 1847.

[12] Cordellier, F. " Transformations de suites scalaires et vectorielles" Thèse de doctorat d'état soutenue à l'université de Lille I, 1981.

[13] Davidon, W.C. "Variable Metric Method for Minimization", $AEC\ research\ Development$, Report ANL−5990, 1959.

[14] Degaichia, H.and Benzine,R, The Wolfe Epsilon Steepest Descent Algorithm ;Vol. 8, 2014, no. 55, 2731 − 2741 .

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

[16] Dennis, J.E. and Moré, JJ. "Quasi-Newton methods, motivation and theory", SIAM .Rev. 19(1977) 46 − 89.

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

[18] Djeghaba, N. and Benzine, R. "Accélération de la convergence de la méthode de la plus forte pente", Demonstratio Mathematica.Vol.39, $N°1(2006)$, $pp.$169 − 181.

[19] Fletcher, R. "A new approch to Variable Metric Algorithms" Computer Journal, 13, $pp.$ 317 − 322, 1970

[20] Fletcher, R, "Practical methods of Optimization,"Second Edition , John Wiley & Sons , Chichester, 1987.

[21] Fletcher, R. "An overview of unconstrained optimization",$in\ Algorithms\ for\ Continuous\ Optimization$: the State of Art, E .Spedicato, ed., Kluwer Academic Publishers, 1994 .

[22] Fletcher, R. and M. Reeves. "Function minimization by conjugate gradients". $Computer\ J.$7, $pp.$149 − 154, 1964.

[23] Fletcher, R. and POWELL, MM., "A rapidly Convergent Descent Method for Minimization", Computer Journal, 6 pp.163 − 168, 1963.

[24] Forsythe G.E. "'On the asymptotic directions of the s-dimentional optimum gradient method,"$Numerische\ Mathematik$ 11, pp. 57 − 76.

[25] Gill, P.E. and Marray, W. "Quasi-Newton Methods for unconstrained optimization,"$J.Inst.\ Maths\ applics$, vol 9, pp 91 − 108, 1972.

[26] Griewank, A., "The global convergence of partitioned BFGS on problems with convex decompositions and Lipschitz gradients", $Math.\ Prog.$ 50 (1991) 141 − 175.

[27] Goldfarb, D., " A Family of Variable Metric Methods Derived by Variational Means," $Mathematics\ of\ Computation$, 24, $pp.$ 23 − 26, 1970.

[28] Goldstein, A. A., and Price, J.F., "An effective Algorithm for Minimization" $Numerische\ Mathematik$, 10, pp. $184 - 189, 1967$.

[29] Nocedal, J. and Wright, S.J. "Numerical Optimization" $Springer.\ Second\ edition.$ 2006.

[30] Mathematics and Mathematical Physics(English Translation), 9pp. 94-112, 1969.

[31] Powell, .M.J.D, “On the convergence of the variable metric algorithms,” $J.Inst.\ Math\ .Appl.$ $7\ (1971), 21 - 36$.

[32] Powell, .M.J.D, “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.

[33] Rahali, N., Djeghaba, N., Benzine, R "Global Convergence of the Armijo Epsilon Steepest Descent Algorithm." $Rev.\ Anal.\ Numer.\ Theor.\ Approx.$ 41 (2012), no. 2, $169 - 180(2013)$

[34] Wolfe, P ."Convergence conditions for ascent méthods". $Siam\ Review$,11,pp.$226 - 235$, (1969).

[35] Wolfe, P ."Convergence conditions for ascent méthods II : Some corrections" $Siam\ Review$, 13, pp.$185 - 188$, (1971).

[36] Wynn, P. "On a device for computing the em(Sn) transformation", $M.T.A.C.$, $10(1956), 91 - 96$.

[37] Wynn, P., "Upon systems of recursions which obtain among quotients of the Padé table," $Numer.Math.$, $8(1966)$ pp. $264 - 269$.

[38] Zoutendijk.G, Nonlinear programming computational methods, in : J. Abadie (Ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970, pp. $37 - 86$.

[39] Y. H. Dai and Y. Yuan (1998), Some properties of a new conjugate gradient method, in : Advances in Nonlinear Programming, ed . Kluwer Academic, Boston, pp. $251 - 262$.