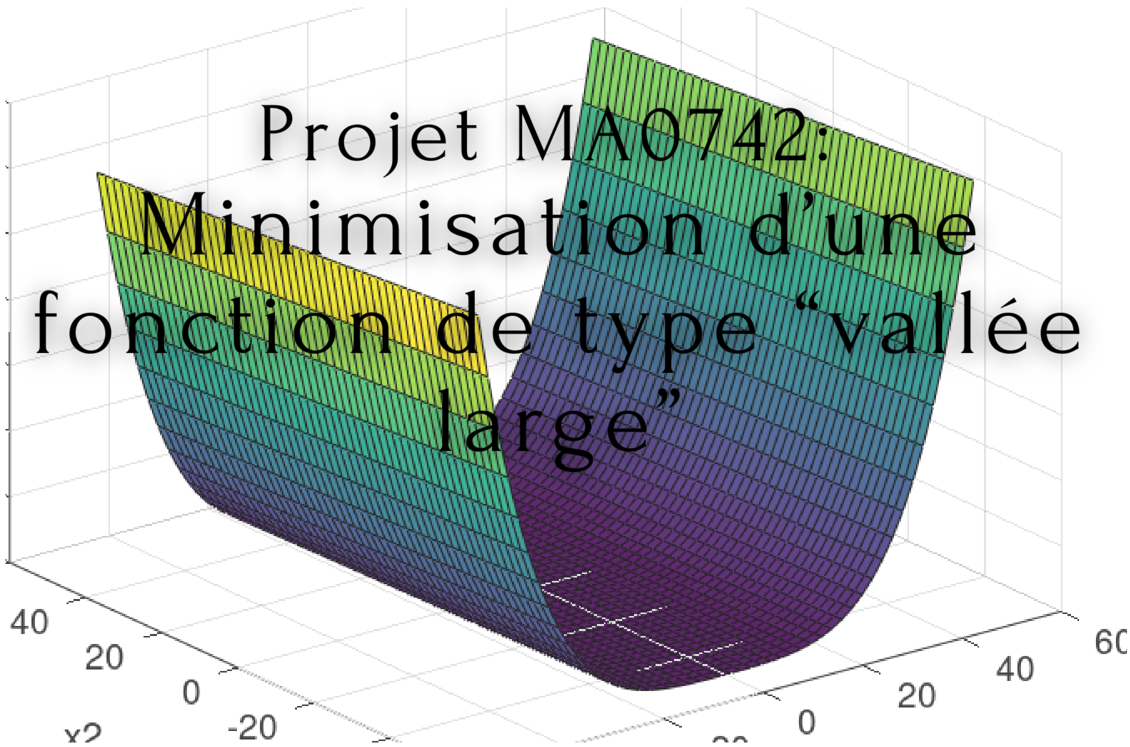
UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

UFR SEN
Sciences Exactes et Naturelles

Projet MA0742: Minimisation d’une fonction de type “vallée large”



PAR
GNAGO YANNICK.

ENSEIGNANTE: LOHRENGEL STÉPHANIE .

2021/2022

SOMMAIRE

Minimisation d'une fonction de type "vallée large"

Objectif :

L'objectif de ce projet est de comparer différentes méthodes vues en cours pour résoudre un problème d'optimisation réputé "difficile". On s'intéresse à la famille de fonctions à deux variables définie par :

$$f_p : x = (x_1, x_2) \in \mathbb{R}^2 \longmapsto f_p(x) = (x_1 - 1)^2 + p(x_1^2 - x_2)^2 \quad (1)$$

Où $p > 0$ est un paramètre réel.

I- Étude mathématique de la fonction :

1. Montrons que $f_p(x_1, x_2) \geq 0$
   $\forall x = (x_1, x_2) \in \mathbb{R}^2$, on a : $(x_1 - 1)^2 \geq 0$ et $p(x_1^2 - x_2)^2 \geq 0$ avec $p > 0$.
   alors $f_p(x) \geq 0$ avec $p > 0$.

2. L'unicité du minimum sur$\mathbb{R}^2$
   On a : $\forall x \in \mathbb{R}^2, f_p \geq 0$ et fp est une fonction polynomiale sur $\mathbb{R}^2$.
   On calcule les dérivées partielles de f_p : $\frac{\partial f}{\partial x_1} = 2(x_1 - 1) + 4px_1(x_1^2 - x_2)$ et $\frac{\partial f}{\partial x_2} = -2p(x_1^2 - x_2)$.
   Si (x, y) est un point critique[1] de f_p, il vérifie : $\begin{cases} \frac{\partial f}{\partial x_1} = 0 \\ \frac{\partial f}{\partial x_2} = 0 \end{cases}$
   Alors : $\begin{cases} 2(x_1 - 1) + 4px_1(x_1^2 - x_2) = 0 \\ -2p(x_1^2 - x_2) = 0 \end{cases} \Longrightarrow \begin{cases} 2(x_1 - 1) = 0 \\ 4p(x_1^2 - x_2) = 0 \end{cases} \Longrightarrow \begin{cases} x_1 = 0 \\ x_2 = 0 \end{cases}$
   Par conséquent $(1, 1)$ est le seul point critique de f_p.
   Comme f_p est différentiable alors les extremums locaux[2]ne peuvent être atteints qu'au point critique, il suffit d'étudier si $(1, 1)$ est un extremum local.
   On a : $\forall (x_1, x_2) \in \mathbb{R}^2$,$f_p(x_1, x_2) \geq 0 = f_p(1, 1)$, Ainsi $x^* = (1, 1)$ est l'unique minimum de f_p

   – x^* depend t-il de p ?
   On a : $x^* = (1, 1)$, donc x^* ne dépend pas du paramètre p.

3. Le gradient et la matrice Hessienne
   On a : $\bigtriangledown f_p(x) = (J_f(x))^t = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2(x_1 - 1) + 4px_1(x_1^2 - x_2) \\ -2p(x_1^2 - x_2) \end{pmatrix}$
   Soit H, la matrice hessienne de f_p, on a : $H = \bigtriangledown^2 f_p(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix}$
   Alors :
   $$H = \begin{pmatrix} 2 + 4p(3x_1 - x_2) & -4px_1 \\ -4px_1 & 2p \end{pmatrix}$$

4. le conditionnement de H en x^*
   On a : $cond(H) = \|H\| \|H^{-1}\|$
   et $H(x^*) = \begin{pmatrix} 2 + 8p & -4p \\ -4p & 2p \end{pmatrix}$ et $H^{-1}(x^*) = \frac{1}{4p} \begin{pmatrix} 2p & 4p \\ 4p & 2 + 8p \end{pmatrix}$
   Comme les 3 normes matricielles sont équivalentes :

1. un point critique d'une fonction de plusieurs variables, à valeurs numériques, est un point d'annulation de son gradient.
2. Un extremum est une valeur extrême, qui peut correspondre à un minimum ou à un maximum, prise par une valeur sur un intervalle donné.

Alors : $\|H\| = \|H(x^*)\|_1 = \max(2+4p, -2p)$ et $\|H^{-1}\| = \|H^{-1}(x^*)\|_1 = \max(\frac{3}{2}, \frac{1}{2p}+3)$
On a : $p > 0 \Longrightarrow 2+4p \geq -2p$ et $\frac{1}{2p}+3 \geq \frac{3}{2}$

$$\text{donc } cond(H(x^*)) \approx (2+4p)(\frac{1}{2p}+3)$$

Comparaison pour plusieurs p
$Si \quad p=1 \Longrightarrow cond(H(x^*)) = 21$
$Si \quad p=3 \Longrightarrow cond(H(x^*)) = 44,33$
$Si \quad p=5 \Longrightarrow cond(H(x^*)) = 68,20$
$Si \quad p=7 \Longrightarrow cond(H(x^*)) = 92,143$
$Si \quad p=10 \Longrightarrow cond(H(x^*)) = 248,05$
Au fur à mesure que p augmente, le conditionnement de la matrice[3] Hessienne en x^* croit de façon exponentielle. Cela montre que le problème de minimisation de f_p est trés mal conditionné lorsque p augmente.

Autre méthode pour montrer l'unicité du minimum

Montrons fp est α-convexe
On a : f_p est une fonction polynomiale sur $\mathbb{R}^2$ alors $f_p \in \mathbb{C}^\infty(\mathbb{R}^2)$.
f est différentiable sur $\mathbb{R}^2$, alors :

$$\bigtriangledown f(x) = (J_f(x))^t = (\frac{\partial f}{\partial x1} \quad \frac{\partial f}{\partial x2})^t$$
$$\bigtriangledown f(x) = \begin{pmatrix} 2(x_1-1)+4px_1(x_1^2-x_2) \\ -2p(x_1^2-x_2) \end{pmatrix}$$

Soit $A = < \bigtriangledown f(x) - \bigtriangledown f(y), y-x > \quad avec \quad x = (x_1 \quad x_2)^t \quad et y = (y_1 \quad y_2)^t$.
On a donc : $\bigtriangledown f(x) - \bigtriangledown f(y) = \begin{pmatrix} 2(y_1-x_1)+4p(y_1(y_1^2-y_2)-x_1(x_1^2-x_2)) \\ 2p(x_2^2-y_1^2+y_2-x_2) \end{pmatrix}$ et $y-x = \begin{pmatrix} y_1-x_1 \\ y_2-x_2 \end{pmatrix}$. Alors :
$A = 2(x_1-y_1)^2 + 4p(y_1-x_1)^2(\frac{y_1(y_1^2-y_2)-x_1(x_1^2-x_2)}{y_1-x_1}) + 2p(y_2-x_2)^2(\frac{x_2^2-y_1^2}{y_2-x_2}+1)$
$A = (x_1-y_1)^2(2+4p\frac{y_1(y_1^2-y_2)-x_1(x_1^2-x_2)}{y_1-x_1}) + (x_2-y_2)^2(2p(\frac{x_2^2-y_1^2}{y_2-x_2}+1))$
Posons : $a = (2+4p\frac{y_1(y_1^2-y_2)-x_1(x_1^2-x_2)}{y_1-x_1}) \quad$ et $\quad b = (2p(\frac{x_2^2-y_1^2}{y_2-x_2}+1))$

$$\text{D'où : } \quad A = a(x_1-y_1)^2 + b(x_2-y_2)^2$$
$$\text{Soit } \alpha = \min(a,b)\,;\text{ on a : } a \geq \alpha \text{ et } b \geq \alpha.$$
$$\Longrightarrow A \geq \alpha(x_1-y_1)^2 + \alpha(x_2-y_2)^2$$
$$\Longrightarrow A \geq \alpha\|x-y\|^2$$

Conclusion : on a f_p est différentiable sur $\mathbb{R}^2$ et $< \bigtriangledown f(x) - \bigtriangledown f(y), y-x > \geq \alpha\|x-y\|^2$.
par conséquent f_p est α-convexe, de plus comme f_p est au moins de classe $\in \mathbb{C}^\infty(\mathbb{R}^2)$ alors d'après le théorème d'existence et d'unicité d'un minimum, f_p admet donc un unique minimum x^* sur $\mathbb{R}^2$.
D'après le théorème de la condition nécessaires d'optimalité(CNO), on sait que :

$$\bigtriangledown f(x^*) = 0 \quad or \quad \bigtriangledown f(x) = \begin{pmatrix} 2(x_1-1)+4px_1(x_1^2-x_2) \\ -2p(x_1^2-x_2) \end{pmatrix}.$$

3. le conditionnement : mesure la dépendance de la solution d'un problème numérique par rapport aux données du problème, ceci afin de contrôler la validité d'une solution calculée par rapport à ces données.

$$\text{On a :}\quad \nabla f(x) = 0 \Longleftrightarrow \begin{pmatrix} 2(\mathrm{x}_1 - 1) + 4px_1(x_1^2 - x_2) = 0 \\ -2p(x_1^2 - x_2) = 0 \end{pmatrix}$$

$$\Longleftrightarrow \begin{pmatrix} 2(\mathrm{x}_1 - 1) = 0 \\ (x_1^2 - x_2) = 0 \end{pmatrix}$$

$$\nabla f(x) = 0 \Longleftrightarrow \begin{pmatrix} \mathrm{x}_1 = 1 \\ x_2 = 1 \end{pmatrix}$$

Donc $x^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$; x^* ne dépend pas du paramètre p..

5. Visualisation de la fonction f_p

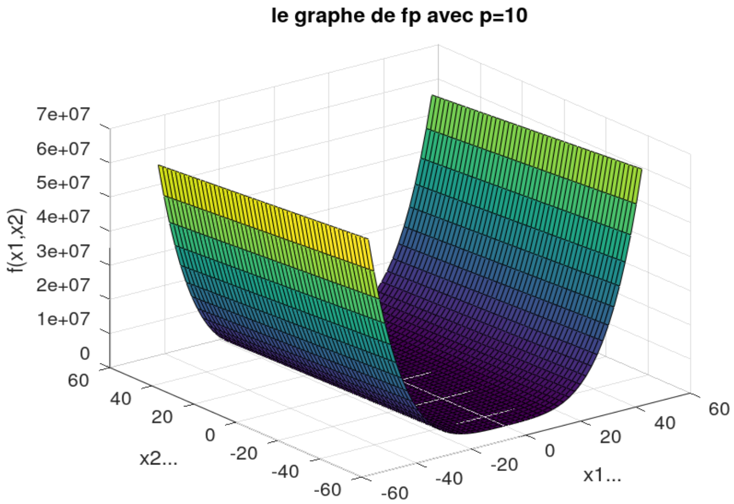


Figure 1

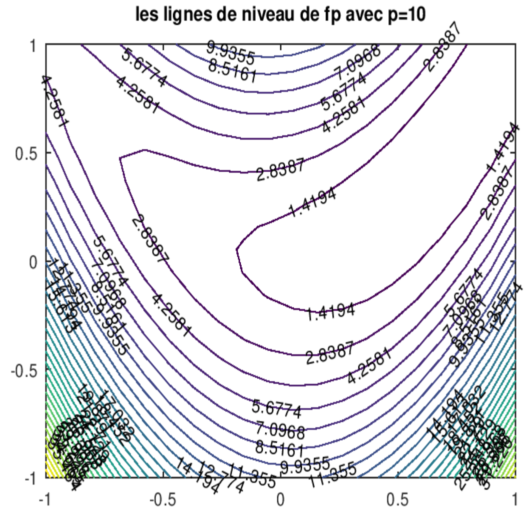


Figure 2

Commentaires :

–Les graphes de f_p ont tous la même forme *i.e* la forme d'une vallée. Il faut noter juste que cette vallée devient de plus en plus grande au fur à mesure que p augmente.

–Les lignes de niveau[4] sont toutes représentées par des ellipses déformées, elles sont presque pareilles aux courbes de niveau sur les cartes topographiques.

(Voir visual.m)

4. Les lignes de niveau reflètent souvent une réalité physique. Sur une carte topographique, elles désignent les points de même altitude.

II- Mise en œuvre : Minimisation sans contraintes

1. La Méthode de gradient

   (a) Principe : Plus connu sous le nom de la descente de gradient, c'est un algorithme d'optimisation qui permet d'avoir le minimum d'une fonction convexe. Elle repose sur la formule :

$$x_{n+1} = x_n - \rho . \bigtriangledown f(x_n)$$

   Concrètement, il s'agit d'avancer tout au long de la fonction en descendant d'un pas ρ. Alors comment trouver le pas ? Le pas détermine la vitesse de la descente. Généralement, on ne choisira pas un gros pas sinon on risque de sauter le minimum donc pour s'assurer de le trouver, on choisira un petit pas.
   (Voir grad.m)

   (b) Visualisation du comportement de la suite

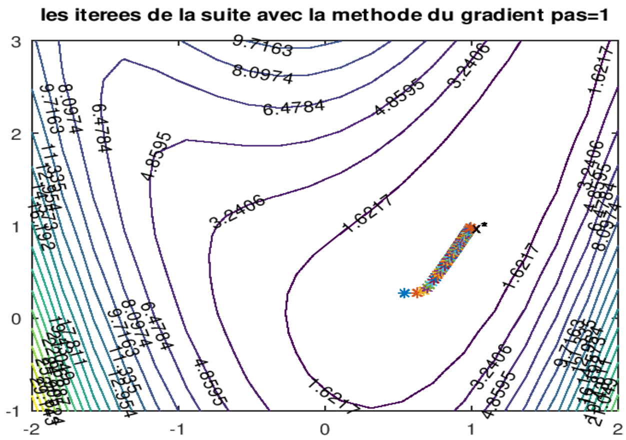


FIGURE 3 – Les itérées avec la méthode du gradient de f_p avec $p = 1, x_0 = [0.4; 0.3]$.

   (c) Commentaires :

   — En fonction de p

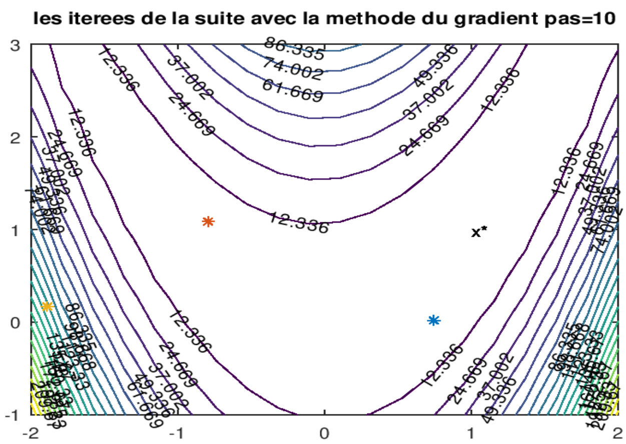


FIGURE 4 – Les itérées avec la méthode du gradient pour p=10, kmax=100, seuil=1e-5, x0=(0.4,0.3), pas=0.1.

On remarque qu'au fur à mesure que p augmente, les itérations s'éloignent considérablement de x^* et le nombre d'itérations diminue. Cela prouve bien la question 4 ; $i.e$ il est donc idéal que p soit petit pour que le problème de minimisation soit bien conditionné.

— <u>En fonction du seuil</u>
Plus le seuil est petit, mieux les itérations semblent converger x^*. Pour un grand seuil, on constate que la méthode ne converge pas vers x^*, ce qui est d'autant normal vu que d'après la CNO [5], $\bigtriangledown f(x^*) = 0$ $i.e$ la norme du gradient de f en x^* doit être proche de 0 .
Voir FIGURE 3

— <u>En fonction de x_0</u>

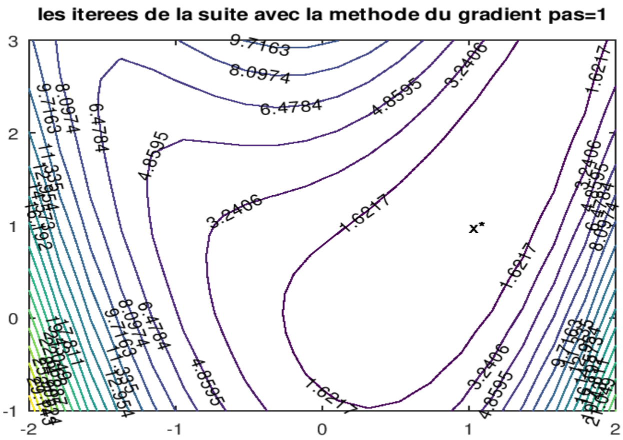


FIGURE 5 – Les itérées avec la méthode du gradient pour p=1, kmax=100, seuil=1e-5, x0=(5 ;8), pas=0.1.

On constate que pour x_0 trés éloigné de x^*, les itérations semblent grossièrement erronées. Par exemple cette figure, la dernière itération est $x = [NaN; Inf]$

— <u>En fonction de ρ (le pas)</u>

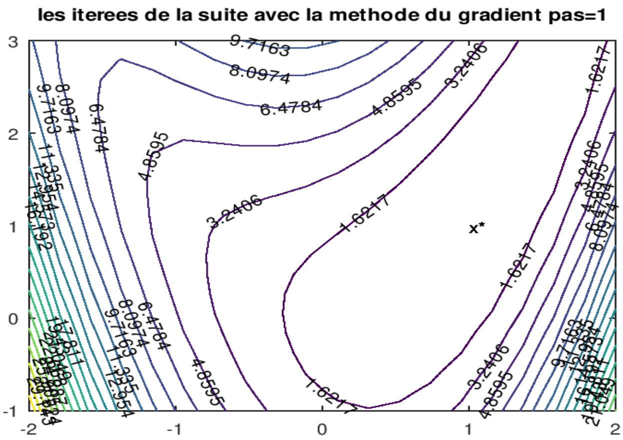


FIGURE 6 – Les itérées avec la méthode du gradient pour p=1, kmax=100, seuil=1e-5, x0=(0.4 ;0.3), pas=11.

Plus ρ est grand, la suite des itérées diverge grossièrement. Il semble donc idéal pour assurer la convergence que le pas soit très petit.

5. CNO : le théorème de la condition nécessaires d'optimalité

2. <u>La Méthode de Newton-Raphson</u>

(a) <u>Principe :</u> L'idée de cette méthode est pareil avec la descente du gradient, mais elle repose sur la formule :

$$x_{n+1} = x_n - H(f(x_n))^{-1}. \bigtriangledown f(x_n)$$

Avec H : La matrice hessienne de f.
(Voir newton.m)

(b) <u>Visualisation pour différentes valeurs de p.</u>
<u>On fixe ici $kmax = 100$, $seuil = 1e-5$ et $x0 = (0;0)$</u>

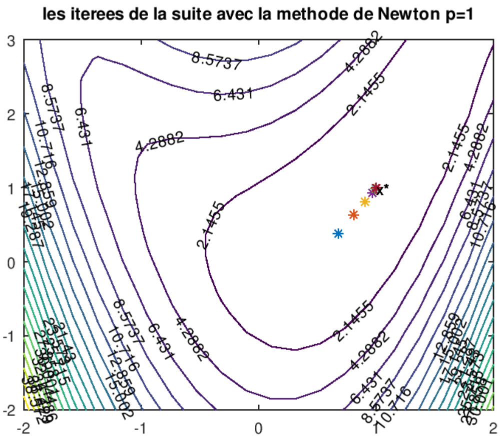


FIGURE 7 – Les itérées avec la méthode de Newton-Raphson pour p=1.

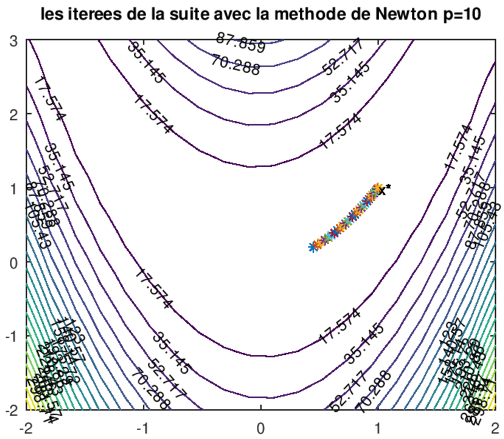


FIGURE 8 – Les itérées avec la méthode de Newton-Raphson pour p=30.

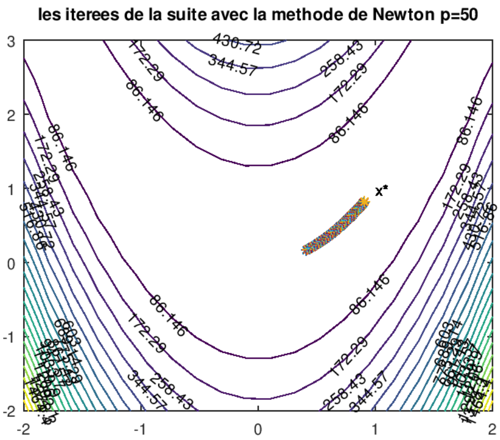


FIGURE 9 – Les itérées avec la méthode de Newton-Raphson pour p=50.

(c) <u>Commentaires :</u>

— <u>En fonction de p :</u>
On constate ici que le paramètre p influence le conditionnement du problème de minimisation avec la méthode de Newton-Raphson n'est pas très large, par exemple :

- Pour $p = 1$, la méthode converge vers x^* au bout de 7 itérations.
- Pour $p = 30$, elle converge vers x^* au bout de 31 itérations.
- Pour $p = 50$, elle ne converge pas vers x^*, mais reste trés proche $x = [0.908; 0.825]$.

Cela montre qu'elle est plus efficace que la méthode du gradient pour un problème de minimisation sur cette famille de fonction à deux variables.

— <u>En fonction de x_0</u>

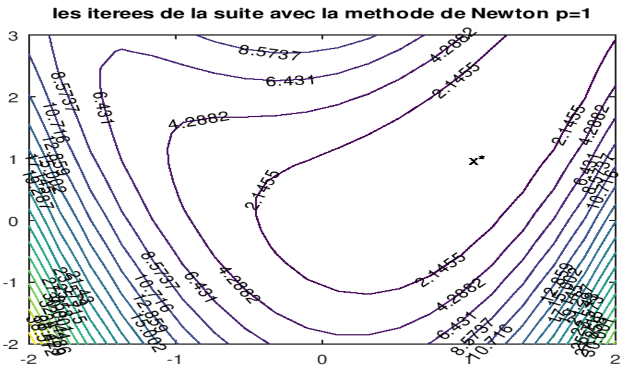


FIGURE 10 – Les itérées avec la méthode de Newton-Raphson pour x0=(5 ;8), p=1, kmax=100, seuil=1e-5.

Plus x_0 est très éloigné de x^*, la méthode diverge grossièrement. Par exemple sur cette figure, la dernière itération est $x = [7.68; 59.04]$.

3. <u>La Méthode de quasi-Newton</u>

   (a) <u>Principe :</u> Plus connu sous le nom de méthode de Broyden-Fletcher-Goldfarb-Shanno (BFGS),le principe de base est le même que la méthode de Newton- Raphson, mais on évite cette fois-ci le calcul de $H(f(x_n))^{-1}$ en la remplaçant par une approximation. (Voir quasi.m)

   (b) <u>Visualisation pour différents paramètres p.</u>
   <u>On fixe ici kmax=100, seuil=1e-5, x0=(0.4 ; 0.3) et $\rho = 0.1(pas)$</u>

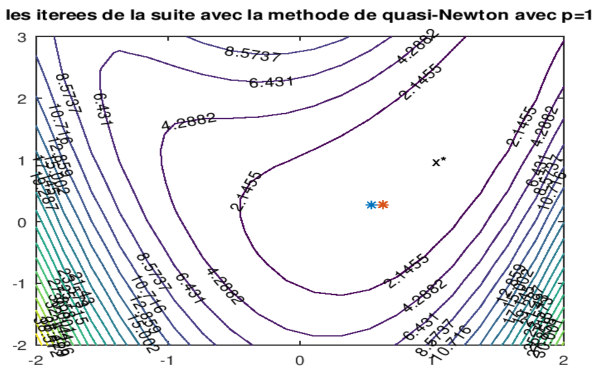


FIGURE 11 – Les itérées avec la méthode de quasi-Newton p=1

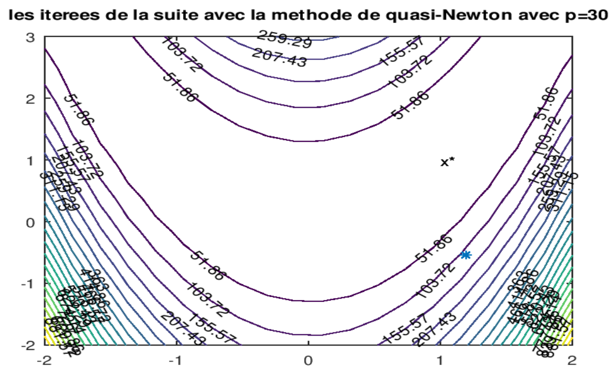


FIGURE 12 – Les itérées avec la méthode de quasi-Newton p=30

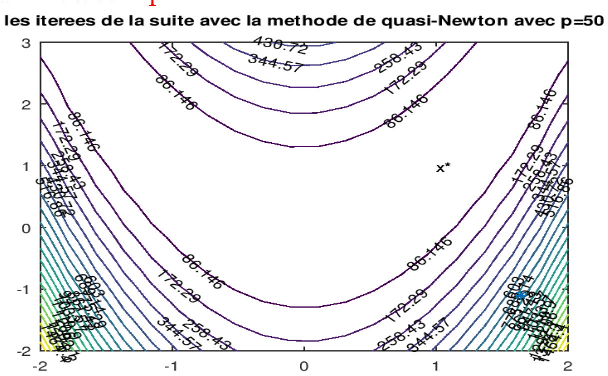


FIGURE 13 – Les itérées avec la méthode de quasi-Newton p=50

(c) Commentaires :

— En fonction de p :
On constate ici que le paramètre p a une influence sur le problème de minimisation avec la méthode de quasi-Newtow, par exemple :

- Pour $p = 1$, la méthode converge vers un point diffèrent de x^* au bout de 3 itérations.
- Pour $p = 30$, elle converge aussi vers un point très diffèrent de x^* au bout de 3 itérations.
- Pour $p = 50$, le constat est pareil.

Nos resultats paraissent suprenants, car la méthode converge bizarrement, ce qui semble d'autant logique, car la méthode de quasi-Newton a consisté a trouver une approximation de l'inverse la matrice Hessienne que nous avons dans la méthode de Newton-Raphson afin de trouver rapidement le minimum vu que la méthode de Newton-Raphson demande assez de calcul.

— En fonction de ρ(le pas fixe) :

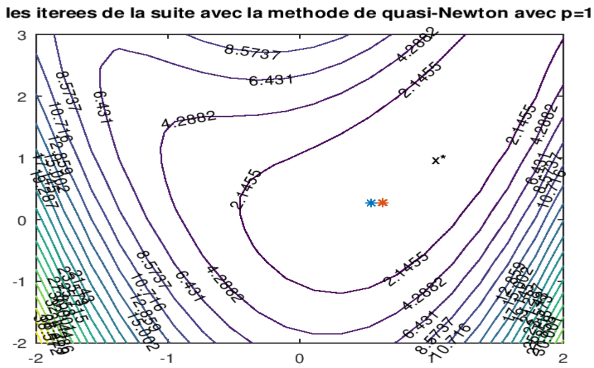


FIGURE 14 – Les itérées avec la méthode de quasi-Newton pour$\rho = 0.1$

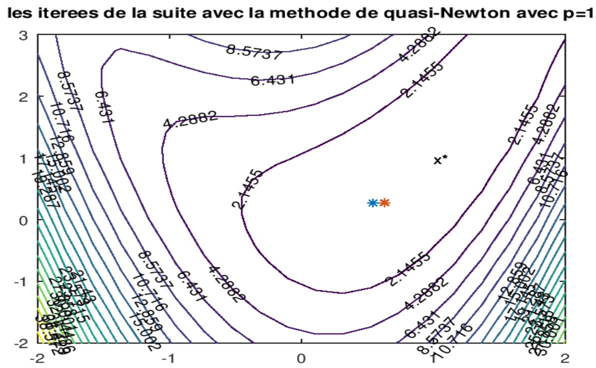


FIGURE 15 – Les itérées avec la méthode de quasi-Newton pour $\rho = 1e - 1$

On constate que pour un pas très petit, la convergence est assurée, mais la dernière itération du minimum reste très loin de x^*.
Ce qui semble aussi normal vu que la méthode de quasi-Newton est une optimisation de Newton-Raphson.

4. La comparaison des méthodes
Pour cette partie, nous avons pris les mêmes paramètres afin d'évaluer chaque méthode.
Nous avons fixé : $p = 1$, $kmax = 100$, $seuil = 1e - 5$, $x0 = [0.4; 0.3]$, $pas = 0.1$.
(Voir compar.m)

- Pour la méthode du gradient, elle ne s'arrête pas, car le nombre d'itérations est 100. L'erreur entre le minimum approché et x^* est trés petit , soit $(7.3217e - 03; 1.7639e - 02)$.
- Pour la méthode de Newton, elle s'arrête au bout de 31 itérations. L'erreur entre le minimum approché et x^* est encore plus petit, soit $(1.8717e - 07; 4.0398e - 07)$.
- Pour la méthode de quasi-Newton, elle s'arrête au bout de 3 itérations. L'erreur entre le minimum approché et x^* est plus grande que l'erreur par la méthode de Newton.

En résumé, je dirai que la méthode de Newton-Raphson semble idéal pour un problème de minimisation sur cette famille de fonctions, mais elle demande assez de calcul comparé à la méthode de quasi-Newton qui ne converge pas très bien, ce qui est d'autant logique, car c'est une approximation de la méthode Newton-Raphson pour éviter le calcul de l'inverse de la matrice hessienne.

Quant à la méthode de gradient, elle semble très correcte pour un problème de minimisation si les paramètres comme le pas ρ sont bien choisis. Ici, dans notre cas, nous avons choisi des paramètres afin que l'erreur soit très petite, mais elle s'exécute moins rapidement, car il faut assez d'itérations pour que le minimum approché se rapproche de x^* .

Calcul du coût total de chaque méthode

En supposant que chaque opération coûte 0.10 euros et que l'évaluation d'une fonction coûte 1 euro, nous avons donc calculé le coût des boucles itératives :

— La méthode de gradient

```
while (test>seuil && k< kmax)                                  Le cout
d=-gradf(x(1),x(2));                                                 1
x=x+pas*d;                              2 x 0.1                    0.2
test=norm(gradf(x(1),x(2))); (2 x(2 x 2 + 1)) x 0.1                0.1
k=k+1;
endwhile
```

$\Longrightarrow$ Le coût total est 1.3 euros.

— La méthode de Newton

```
while (test>seuil && k< kmax)                                  Le cout
H=Hessf(x(1),x(2));                                                  1
d=H\r;                       2 x 2 x 2 x 0.1                       0.8
x=x+d;                       2 x 0.1                               0.2
r=-gradf(x(1),x(2));                                                 1
test=norm(r,2);             (2 x(2 x 2 + 1)) x 0.1                   1
k=k+1;
endwhile
```

$\Longrightarrow$ Le coût total est 4 euros.

— La méthode de quasi-Newtion

```
while (test>seuil && k< kmax)                                  Le cout
x = x + pas*d ;                    2 x 0.1                         0.2
r=-gradf(x(1),x(2));                                                 1
d=M*r;                     (2 x 2 x 2 -2)x 0 .1                    0.6
% Calcul de M
s=pas*d ;                          2 x 0.1                         0.2
y=gradf(x(1),x(2))+r ;             2 x 0.1                         0.2
S=eye(2)-((s*y')/dot(s,y)) ;
M=S*M*S'+((s*s')/dot(s,y));
test=norm(r,2);            (2 x(2 x 2 + 1)) x 0.1                    1
k=k+1;
endwhile
```

$\Longrightarrow$ Le coût total est 3,2 euros.

III- La règle de Wolfe

Références

[1] *Optimisation et analyse convexe*, **de Jean-Baptiste Hiriart-Urruty.**

[2] *Convex analysis and minimization algorithms*, **, de Jean-Baptiste Hiriart-Urruty et Claude Lemarechal.**

[3] *Méthodes numériques pour l'ingénieur : utilisation de l'outil MATLAB : cours, exercices et problèmes de synthèse corrigés*, **d'Abdelkhalak El Hami et Bouchaïb Radi.**

[4] Exercices corrigés - Extrema des fonctions de plusieurs variables.

[5] Algorithme du gradient.

[6] Méthode de Broyden-Fletcher-Goldfarb-Shanno

[7] Méthodes quasi-Newton.

[8] Conditionnement (analyse numérique).