Stochastic Optimization

done by

Jean-Claude S. MITCHOZOUNOU

Africa Business School, UM6P, Morocco

Master of Quantitative and Financial Modelling (QFM)

Practical work

Teacher: Prof. Abdeslam Kadrani

Academic Year 2023-2024

CONTENTS

TP3 Pénalités extérieures

0.1 Etude théorique: Optimisation avec contraintes

Considérons le problème d'optimisation avec contraintes suivant :

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

avec $X = \{x \in \mathbb{R}^2 : x_1 \leq -\frac{1}{2}, x_2 \leq -\frac{1}{2}\}$ et $f : \mathbb{R}^2 \to \mathbb{R}$ définie par :

$$f(x) = 2x_1^2 + 3x_1x_2 + 2x_2^2$$

1.a) Solution du probléme de minimisation de f sans les contraintes de X.
Il s'agit de résoudre le problème

$$(P_1) \qquad \min_{x \in \mathbb{R}^{\not{2}}} f(x)$$

f est une fonction polynomiale, elle est donc de classe $\mathcal{C}^\infty(\mathbb{R}^2)$. Alors $\forall\ x = (x_1, x_2) \in \mathbb{R}^2$, on a:

$$\nabla f(x) = (4x_1 + 3x_2, 4x_2 + 3x_1)$$

. Soit $x^+ = (x_1^+, x_2^+)$ une solution de (P_1). D'après l'équation d'Euler on déduis que $\nabla f(x^+) = 0$.

$$\nabla f(x^+) = 0 \iff \begin{cases} 4x_1^+ + 3x_2^+ = 0 \\ 4x_2^+ + 3x_1^+ = 0 \end{cases}$$
$$\iff \begin{cases} 7x_1^+ = 0 \\ 7x_2^+ = 0 \end{cases}$$
$$\iff x_1^+ = x_2^+ = 0$$

Ainsi le problème (P_1) a pour solution $\boxed{x^+ = (0,0)}$

1.b) Soit x^* solution du problème de minimisation de f avec les contraintes de X, i.e.

$$x^* \in arg \min_{x \in X} f(x)$$

Démontrons que nécéssairement, $\nabla_x \mathcal{L}(x^*, \mu) = 0$ avec $x^* \in \partial X$ où ∂X désigne la frontière de X et $\mu \in \mathbb{R}$.
Le problème (P) est équivalent à:

D'où la preuve.

$\min_x f(x)$
$sc.$

$$\begin{cases} g_1(x) \leq 0 \\ g_2(x) \leq 0 \end{cases} \quad \text{avec } g_1(x) = x_1 + \tfrac{1}{2}, \text{ et } g_2(x) = x_2 + \tfrac{1}{2}$$

f, g_1, g_2 sont différentiables. De plus f est convexe car $H_{ess}f(x) = \begin{pmatrix} 4 & 3 \\ 3 & 4 \end{pmatrix}$ est définie positive.
Les contraintes sont convexes car g_1, g_2 sont affines et l'intérieur de X definie par $int(X) = \{x \in \mathbb{R}^2 : x_1 < -\frac{1}{2}, x_2 < -\frac{1}{2}\}$ est non vide.
Par suite les contraintes de qualifications sont satisfaites

Posons:

$$\mathcal{L}(x, \mu) = f(x) + \mu_1 g_1(x) + \mu_2 g_2(x), \text{ où } \mu = (\mu_1, \mu_2) \in \mathbb{R}^2$$

. $\mathcal{L}$ est bien différentiable et on a:

$$\nabla_x \mathcal{L}(x, \mu) = \nabla f(x) + \mu_1 \nabla g_1(x) + \mu_2 \nabla g_2(x)$$

Puisque la famille $\{\nabla g_1(x), \nabla g_2(x)\} = \{(1,0),(0,1)\}$ est libre alors d'apres le théorème de Karush-Kuhn-Tucker, x^* vérifie:

$$\begin{cases} \nabla f(x^*) + \mu_1 \nabla g_1(x^*) + \mu_2 \nabla g_2(x^*) = 0 \\ \mu_1 g_1(x^*) = 0, \ \mu_1 \in \mathbb{R} \\ \mu_2 g_2(x^*) = 0, \ \mu_2 \in \mathbb{R} \\ g_1(x^*) \leq 0 \\ g_1(x^*) \leq 0 \end{cases} \qquad (*)$$

$$(*) \iff \begin{cases} 4x_1^* + 3x_2^* + \mu_1 = 0 \quad (a) \\ 4x_2^* + 3x_1^* + \mu_2 = 0 \quad (b) \\ \mu_1(x_1^* + \frac{1}{2}) = 0, \ \mu_1 \in \mathbb{R} \quad (c) \\ \mu_2(x_2^* + \frac{1}{2}) = 0, \ \mu_2 \in \mathbb{R} \quad (d) \\ x_1^* + \frac{1}{2} \leq 0 \quad (e) \\ x_2^* + \frac{1}{2} \leq 0 \quad (f) \end{cases}$$

$$(c) \ \text{ et } \ (d) \iff \begin{cases} \mu_1 = 0 \text{ ou } x_1^* = -\frac{1}{2} \\ \mu_2 = 0 \text{ ou } x_2^* = -\frac{1}{2} \end{cases}$$

$-$ Si $\mu_1 = 0$ alors $4x_1^* + 3x_2^* = 0 \iff 4x_1^* = -3x_2^*$ ①. ① dans ⓑ donne $x_2^+ = -\frac{4}{7}\mu_2, \ \mu_2 \in \mathbb{R}$.
Ainsi pour $\mu_2 = -7$ on obtient $x_2^* = 4, x_1^* = -3$ et $(x_1^*, x_2^*) = (-3, 4) \in X$. Par suite $x_1^* = -\frac{1}{2}$
$-$De manière analogue on montre que si $\mu_2 = 0$, il y a contradiction et donc forcement $x_2^* = -\frac{1}{2}$.
De tout ce qui précède, on déduit que $x^* = (x_1^*, x_2^*) = (-\frac{1}{2}, -\frac{1}{2})$ avec $\mu_1 = \mu_2 = -\frac{1}{2}$. ⓢ
Par ailleurs $\partial X = Adh(X) \backslash int(X)$ or $Adh(X) = X$ car X est un fermé pour la topologie naturellle sur $\mathbb{R}^2$ et $int(X) = \{x \in \mathbb{R}^2 : x_1 < -\frac{1}{2}, x_2 < -\frac{1}{2}\}$. Donc on déduit que :
$\partial X = ([-\infty, -\frac{1}{2}])^2 \backslash (]-\infty, -\frac{1}{2}[)^2 = \left\{(-\frac{1}{2}, -\frac{1}{2})\right\}$. $\boxed{\partial X = \left\{(-\dfrac{1}{2}, -\dfrac{1}{2})\right\}}$ ⓣ.

De ⓢ et ⓣ on conclut que que $x^* \in \partial X$. D'où le résultat.

0.2 Etude Numérique: Implantation dans Scilab

Considérer la pénalité extérieure sur l'ensemble X. La fonction pénalisée $\varphi(.,\rho)$ s'écrit:

$$\varphi(.,\rho) : x \to \varphi(x,\rho) = f(x) + \frac{1}{2}\rho P(x)$$

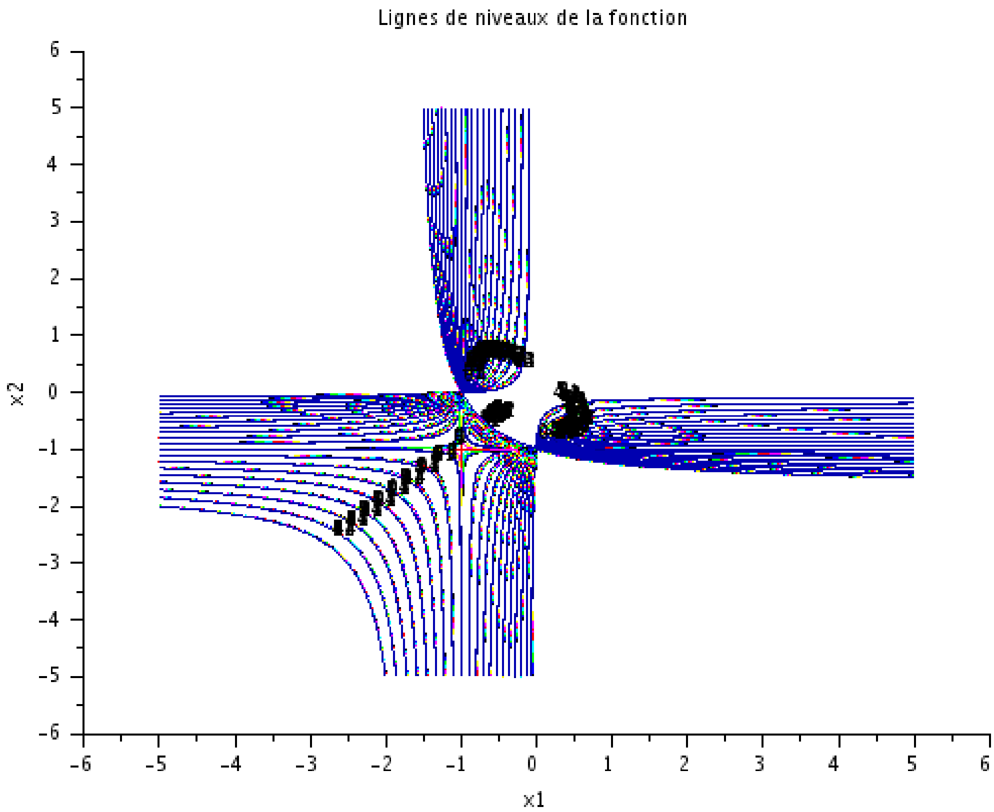
2.a) Déterminer le gradient $\nabla_x \varphi(.,\rho)$ de la fonction pénalisée $\varphi(.,\rho)$.
$P(x) = (g_1^+(x))^2 + (g_2^+(x))^2 = (\max 0, x_1 + \frac{1}{2})^2 + (\max 0, x_2 + \frac{1}{2})^2$. Alors $\varphi(x,\rho)$ s'écrit:

$$\varphi(x,\rho) = \begin{cases} f(x) \text{ si } x \in X \\ f(x) + \frac{1}{2}\rho\left[(x_1 + \frac{1}{2})^2 + (x_2 + \frac{1}{2})^2\right] \text{ sinon} \end{cases}$$

Par suite, $\boxed{\nabla_x \varphi(.,\rho) : x \to \nabla \varphi(x,\rho) = \begin{cases} \nabla f(x) \text{ si } x \in X \\ \nabla f(x) + \rho\left[x_1 + \frac{1}{2}, x_2 + \frac{1}{2}\right] \text{ sinon} \end{cases}}$

2.b) Traçons les courbes de niveaux de la nouvelle fonction $\varphi(.,\rho)$ et son gradient $\nabla_x \varphi(.,\rho)$.

Lignes de niveaux du gradient de $\varphi(.,\rho)$

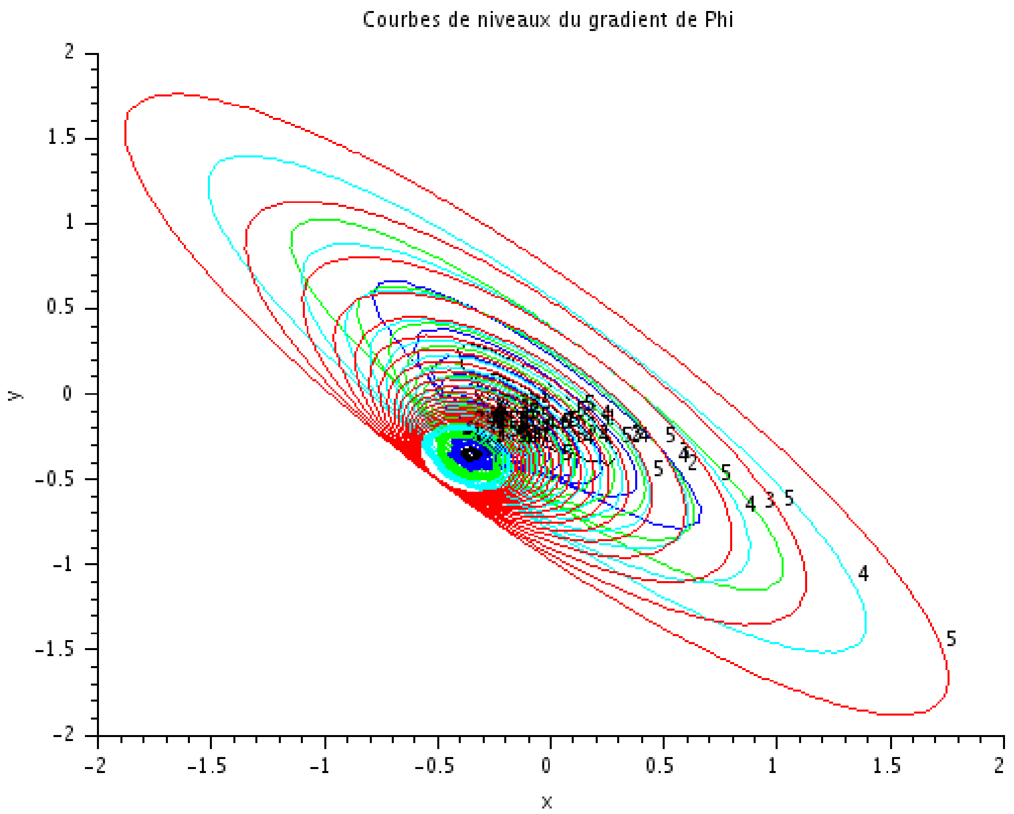


Code Scilab du tracé(Exécutez le fichier *Lignes_ Niveaux.sce*)

```
// Définissons la fonction f et les paramètres
function f = f(x, y)
    f = 2*x^2 + 3*x*y + 2*y^2; // Exemple de fonction
```

```
endfunction


rho = linspace(10, 1000, 20); // Valeurs de rho


// La fonction phi

function fi = Phi(x,y, ho)
    if (x<=-0.5) & (y<=-0.5) then
        fval = f(x,y);
        fi = fval;
    else
        fval = f(x,y);
        fi = fval + 0.5*ho*[(x+0.5)^2 + (y+0.5)^2];
    end
endfunction



// Les lignes de niveaux
x1 = linspace(-5, 5, 100); // Plage de valeurs pour x1
x2 = linspace(-5, 5, 100); // Plage de valeurs pour x2
[X, Y] = ndgrid(x1, x2); // Grille

for ho = 1:length(rho)

    Z = Phi(X,Y,ho);

    contour(x1, x2, Z, [-3:3]);
    xlabel('x1');
    ylabel('x2');
    title('Lignes de niveaux de la function Varphi');
end

```

Courbes de niveaux du gradient de $\nabla_x \varphi(., \rho)$

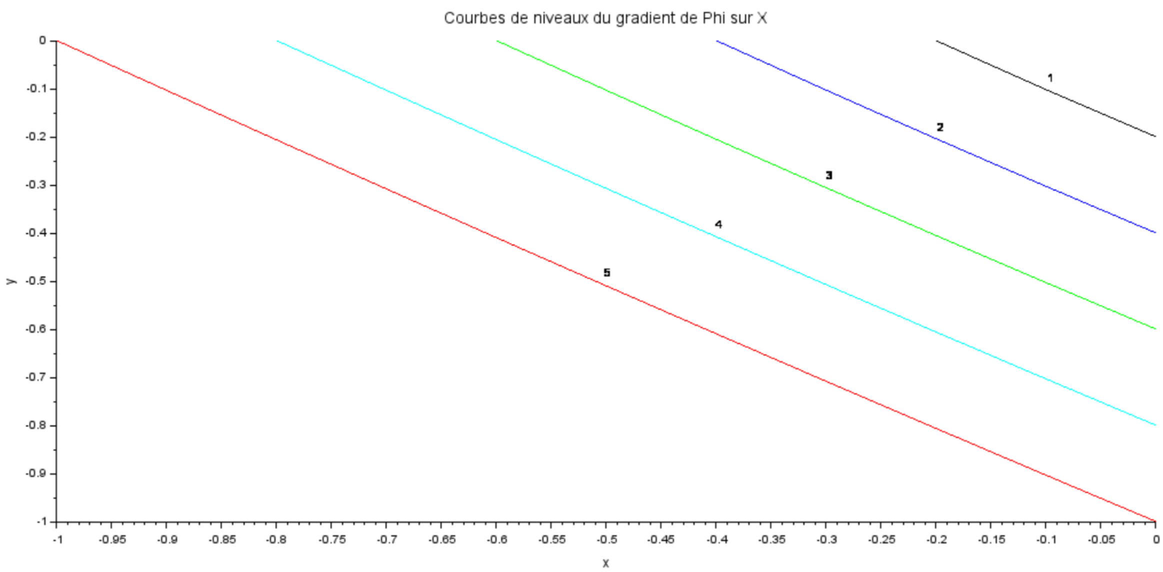


Code scilab du tracé(Exécutez le fichier (*Courbes_NiveauxGrad.sce*)

```
    // La fonction f(x, y)

function f = f(x, y)
    f = 2*x^2 + 3*x*y + 2*y^2; // Exemple de fonction
endfunction

// La fonction phi

function fi = Phi(x,y, ho)
    if (x<=-0.5) & (y<=-0.5) then
        fval = f(x,y);
        fi = fval;
    else
        fval = f(x,y);
        fi = fval + 0.5*ho*[(x+0.5)^2 + (y+0.5)^2];
    end
endfunction



// Calculer les composantes du gradient de phi(x, y)

function [fx,fy] = Grad_Phi(x,y, ho)
    if (x<=-0.5) & (y<=-0.5) then
        fx = 4*x + 3*y;
        fy = 4*y + 3*x;
```

```
    else
        fx = 4*x + 3*y+ ho*(x+0.5);
        fy = 4*y + 3*x + ho*(y+0.5);
    end
endfunction


rho = linspace(10, 1000, 20); // Valeurs de rho

// Plage de valeurs pour x et y
x = linspace(-5, 5, 100);
y = linspace(-5, 5, 100);

// Créer une grille de valeurs pour x et y
[X, Y] = ndgrid(x, y);


for ho = 1:length(rho)
    // Calcul des composantes du gradient pour chaque point de la grille
    [grad_x, grad_y] = Grad_Phi(X, Y,ho);

    // Tracer les courbes de niveaux du gradient
    contour(x, y, sqrt(grad_x.^2 + grad_y.^2),[-3:4]);
    xlabel('x');
    ylabel('y');
    title('Courbes de niveaux du gradient de Phi sur X');
end
```

⋆ **Traçons le gradient de $\varphi(.,\rho)$ uniquement sur le domaine admissible X**



Code Scilab du tracé(Exécutez le fichier($Courbes\_Niveaux\_surX.sce$)

```
// La fonction f(x, y)

function f = f(x, y)
    f = 2*x^2 + 3*x*y + 2*y^2; // Exemple de fonction
```

```
endfunction

// La fonction phi sur X

function fi = Phi(x,y, ho)
    fval = f(x,y);
    fi = fval + 0.5*ho*[(x+0.5)^2 + (y+0.5)^2];
endfunction



// Calculer les composantes du gradient de phi(x, y)

function [fx,fy] = Grad_Phi(x,y, ho)
     fx = 4*x + 3*y+ ho*(x+0.5);
     fy = 4*y + 3*x + ho*(y+0.5);
endfunction


rho = linspace(10, 1000, 20); // Valeurs de rho

// Plage de valeurs pour x et y dans le domaine X
x = linspace(-5, 0, 100);
y = linspace(-5, 0, 100);

// Créer une grille de valeurs pour x et y
[X, Y] = ndgrid(x, y);


for ho = 1:length(rho)
    // Calcule des composantes du gradient pour chaque point de la grille
    [grad_x, grad_y] = Grad_Phi(X, Y,ho);

    // Tracer les courbes de niveaux du gradient
    contour(x, y, sqrt(grad_x.^2 + grad_y.^2),[-3:4]);
    xlabel('x');
    ylabel('y');
    title('Courbes de niveaux du gradient de Phi sur X');
end
```

2.c) Exécutez le fichier testJC.sce qui fait appel au fichier Algo_PénalitéExtérieur.sce

⋆ **Test pour** $\rho = 10^4$ **et** $c = 10$

```
-->exec('/home/jcler/UM6P_QFM_S2/Stochastic_Optimization/TP3_Stochastique/testJC.sce', -1)
Warning : redefining function: Algo_PénalitéExtérieur  . Use funcprot(0) to avoid this message
    iter     rhok       xk.
      0      10000.0   -3.000e-01    5.000e-01
      1     100000.0   -4.997e-01   -4.997e-01

X optimal -4.997e-01  -4.997e-01
```

⋆ **Test pour** $\rho = 10^4$ **et** $c = 2$

```
-->exec('/home/jc1er/UM6P_QFM_S2/Stochastic_Optimization/TP3_Stochastique/testJC.sce', -1)
Warning : redefining function: Algo_PénalitéExtérieur  . Use funcprot(0) to avoid this message
    iter    rhok      xk.
      0     10000.0  -3.000e-01   5.000e-01
      1    100000.0  -4.997e-01  -4.997e-01

X optimal -4.997e-01  -4.997e-01
```

Appréciation la vitesse de convergence pour $\rho = 10^4$: Pour un nombre maxiamal de 20 itération on rematrque que la méthode converge en une seule itération, on conclut donc que la vitesse de convergence est satisfisante

⋆ **Test pour** $\rho = 1$ **et** $c = 10$

```
-->exec('/home/jc1er/UM6P_QFM_S2/Stochastic_Optimization/TP3_Stochastique/testJC.sce', -1)
Warning : redefining function: Algo_PénalitéExtérieur  . Use funcprot(0) to avoid this message
    iter    rhok      xk.
      0         1.0  -3.000e-01   5.000e-01
      1        10.0  -6.250e-02  -6.250e-02

      2       100.0  -2.941e-01  -2.941e-01

      3      1000.0  -4.673e-01  -4.673e-01

      4     10000.0  -4.965e-01  -4.965e-01

      5    100000.0  -4.997e-01  -4.997e-01

X optimal -4.997e-01  -4.997e-01
```

⋆ **Test pour** $\rho = 1$ **et** $c = 2$

```
Warning : redefining function: Algo_PénalitéExtérieur  . Use funcprot(0) to avoid this message
    iter    rhok      xk.
      0         1.0  -3.000e-01   5.000e-01
      1         2.0  -6.250e-02  -6.250e-02

      2         4.0  -1.111e-01  -1.111e-01

      3         8.0  -1.818e-01  -1.818e-01

      4        16.0  -2.667e-01  -2.667e-01

      5        32.0  -3.478e-01  -3.478e-01

      6        64.0  -4.103e-01  -4.103e-01

      7       128.0  -4.507e-01  -4.507e-01

      8       256.0  -4.741e-01  -4.741e-01

      9       512.0  -4.867e-01  -4.867e-01

     10      1024.0  -4.933e-01  -4.933e-01

     11      2048.0  -4.966e-01  -4.966e-01

     12      4096.0  -4.983e-01  -4.983e-01

     13      8192.0  -4.991e-01  -4.991e-01

     14     16384.0  -4.996e-01  -4.996e-01

X optimal -4.996e-01  -4.996e-01
```

Appréciation la vitesse de convergence pour $\rho = 1$: Pour un nombre maxiamal de 20 itération on rematrque que la méthode converge en 5 itérations pour $c = 10$ et en 14 itérations pour $c = 2$,la méthode converge donc lentement; on conclut donc que la vitesse de convergence n'est satisfisante

Conclusion: La méthode converge plus vite lorsque ρ prend une grande valeur tandis qu'elle converge lentement lorsque ρ est petit.
La solution trouvée trouvée est bien très approximative de celle obtenue par la méthode de KKT.