

UNIVERSITÉ DE LILLE
M1 INFORMATIQUE - MACHINE LEARNING

Rendu Atelier #2
Descente de gradient
(quasi-)Newton methode (Ordre 1 + Ordre 2)

Selim Lakhdar

Phillipe Lesueur

16 avril 2021

1 Les fonctions

Les fonctions étudiées sont toutes proportionnelles au paramètre α et elles s'annulent toutes au point 0.

1.1 Fonction 1

$$f_1^\alpha(x) = \alpha \sum_{i=1}^n x_i^2 \quad (1.1)$$

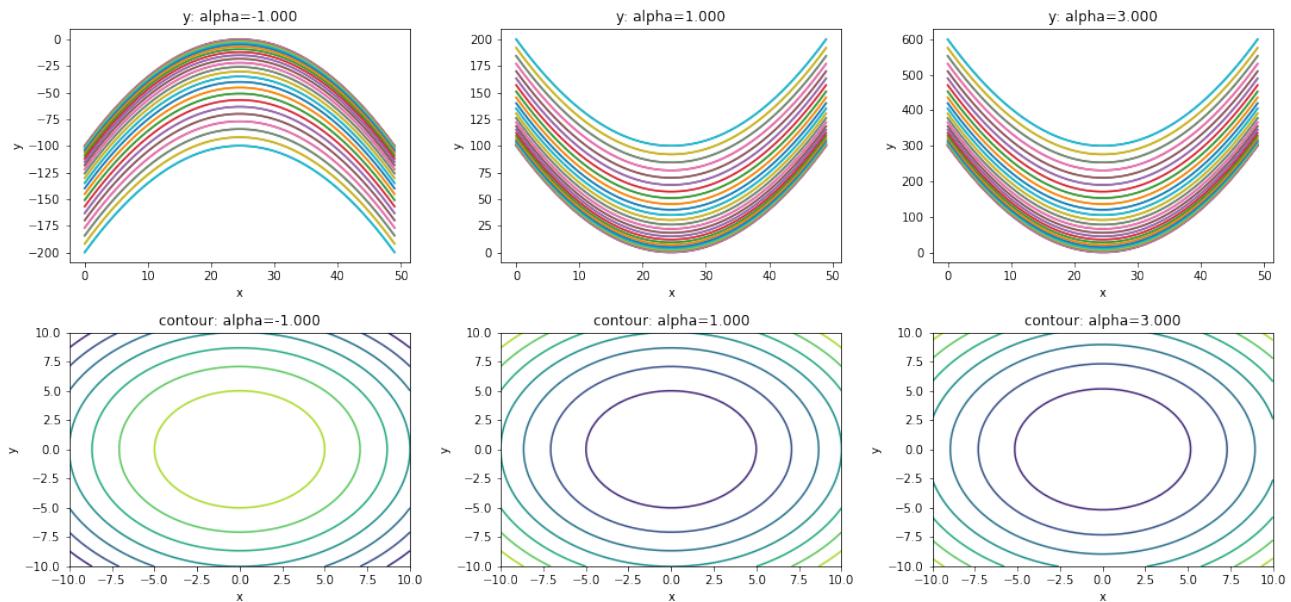


FIGURE 1.1 – $f_1^\alpha(x)$ y

1.2 Fonction 2

$$f_2(x) = \frac{1}{2} \sum_{i=1}^n 10^{\alpha \frac{i-1}{n-1}} \cdot x_i^2 \quad (1.2)$$

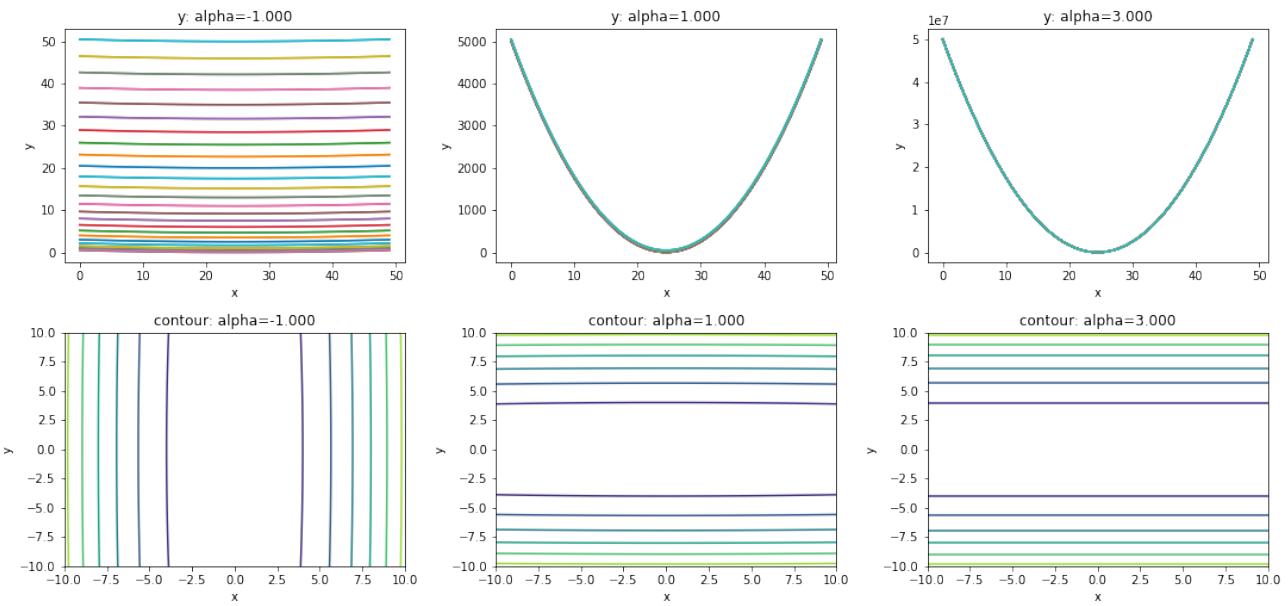


FIGURE 1.2 – $f_1^\alpha(x)$ y

1.3 Fonction 3

$$f_3(x) = 10.n + \sum_{i=1}^n (10^{\alpha \frac{i-1}{n-1}} \cdot (x_i - 1)^2 - 10 \cdot \cos(2\pi(x_i - 1))) \quad (1.3)$$

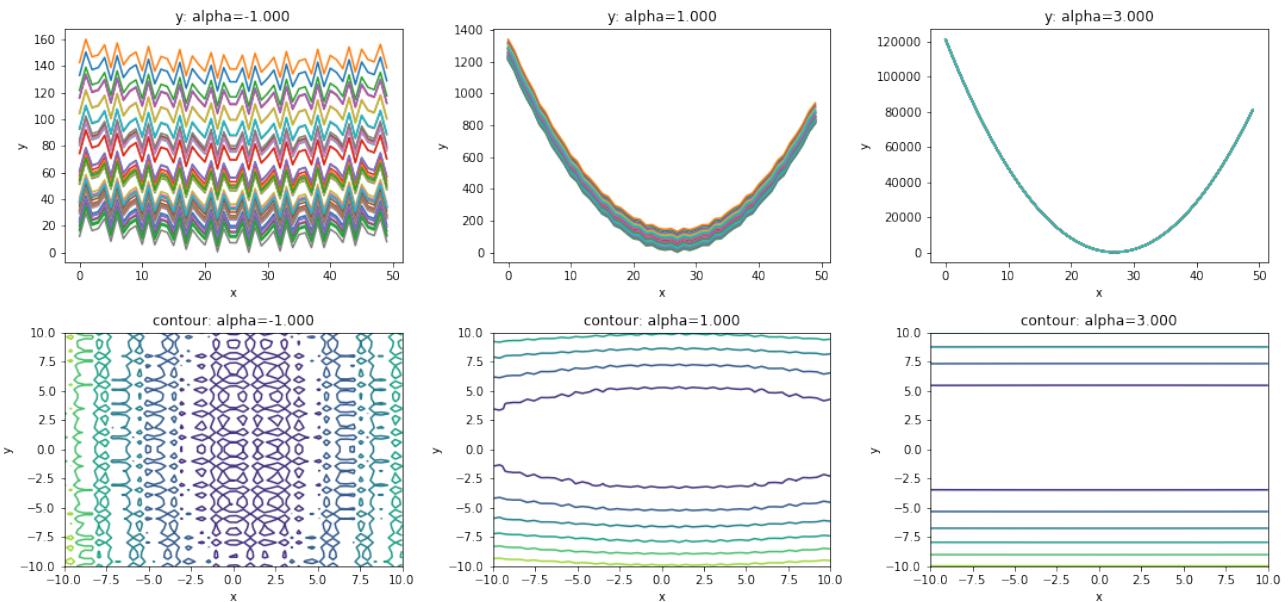


FIGURE 1.3 – $f_3^\alpha(x)$ y

1.4 Fonction 4

$$f_4(x) = \sum_{i=1}^n (10^\alpha \cdot ((x_i - 1)^2 - (x_{i+1} - 1)^2 + (x_i - 2)^2)) \quad (1.4)$$

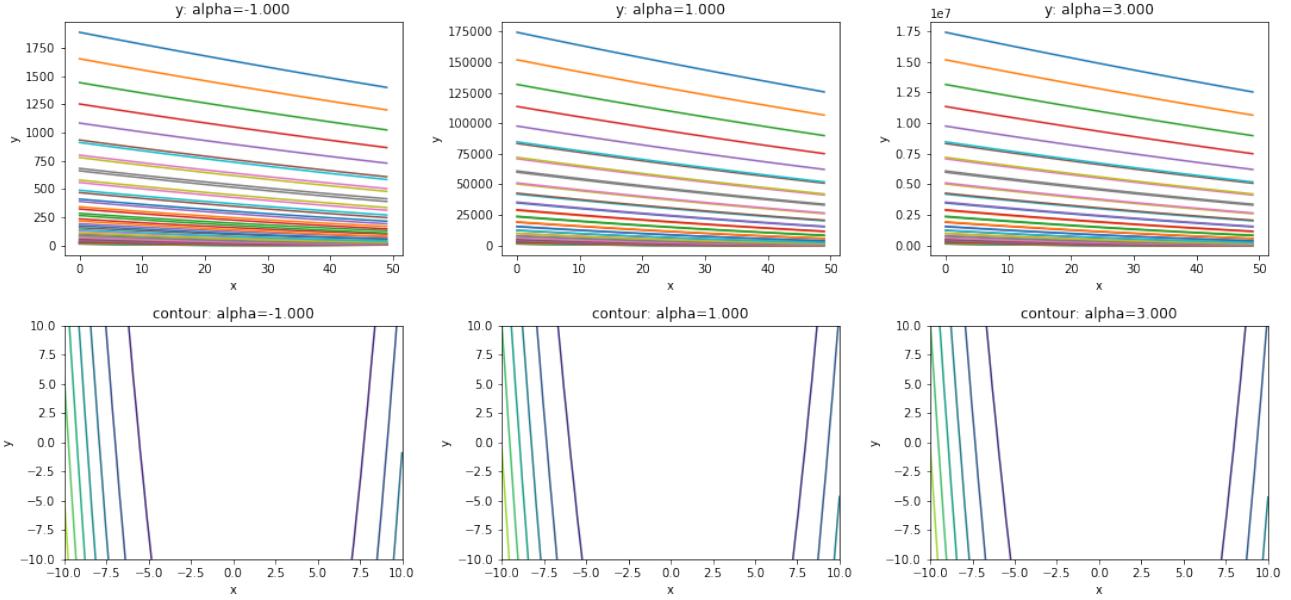


FIGURE 1.4 – $f_4^\alpha(x)$ y

2 Gradient Descent : Fixed Step Size

Dans cette partie nous allons observer une descente de gradient avec un step size fixé. Pour cela nous devons calculer les dérivées de chaque fonctions :

$$f_1'^\alpha(x) = \alpha \sum_{i=1}^n 2x_i \quad (2.1)$$

$$f_2'(x) = \sum_{i=1}^n 10^{\alpha \frac{i-1}{n-1}} \cdot x_i \quad (2.2)$$

$$f_3'(x) = \sum_{i=1}^n (10^{\alpha \frac{i-1}{n-1}} 2 \cdot (x_i - 1) + 10.2\pi \sin(2\pi(x_i - 1))) \quad (2.3)$$

$$f_4'(x) = \begin{cases} \sum_{i=1}^n 4 \cdot 10^\alpha \cdot (x_i - 1)((x_i - 1)^2 - (x_{i+1} - 1)^2) + 2(x_i - 2) & x = 0 \\ \sum_{i=1}^n 4 \cdot 10^\alpha \cdot (x_i - 1)((x_i - 1)^2 - (x_{i+1} - 1)^2) + 2(x_i - 2) + \sum_{i=1}^n -2 \cdot 10^\alpha ((x_i - 1)^2 - (x_{i+1} - 1)) & 0 < x < n \\ \sum_{i=1}^n -2 \cdot 10^\alpha ((x_i - 1)^2 - (x_{i+1} - 1)) & x = n \end{cases} \quad (2.4)$$

Par la suite nous allons observer l'évolution des X pendant la descente du gradient. Ainsi que les différentes valeurs de Y. Nous avons choisi d'exposer ici les variantes avec les valeurs min et max de alpha 0.005 et 3, vous retrouverez dans le notebook l'intégralité des observations.

2.1 Fonction 1

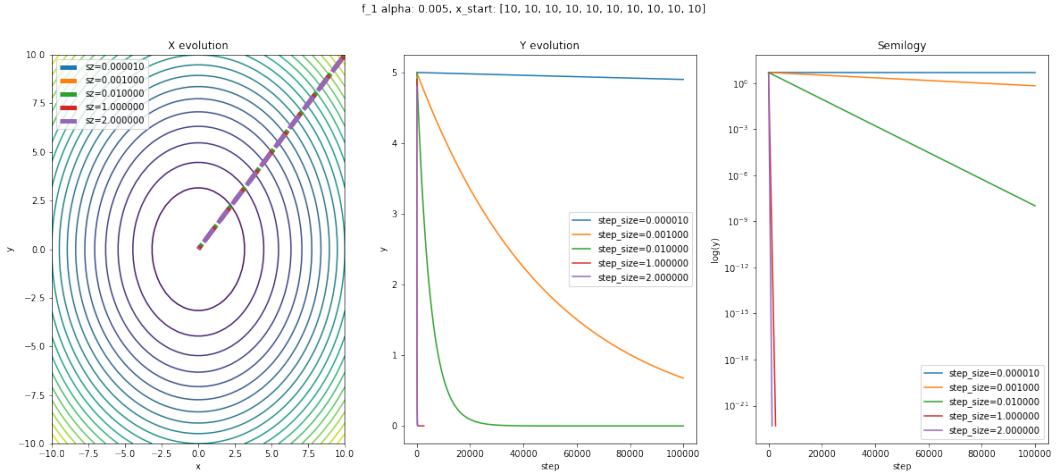


FIGURE 2.1

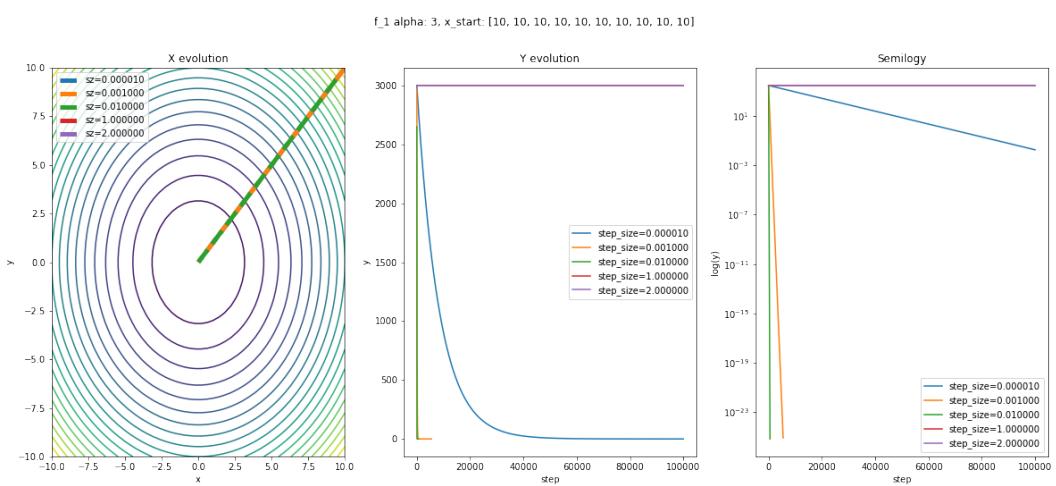


FIGURE 2.2

- La fonction $f_1^\alpha(x)$ reste assez simple (fonction convexe) ce qui permet de converger assez simplement.
- Même en complexifiant la fonction (ie : en prenant une grande valeur de α) on arrive à converger.
- Cependant on remarque que dans certains cas il nous faut plusieurs itérations.
- Cela peut être expliqué par des sauts trop grand ou trop petit à cause du *step_size*

2.2 Fonction 2

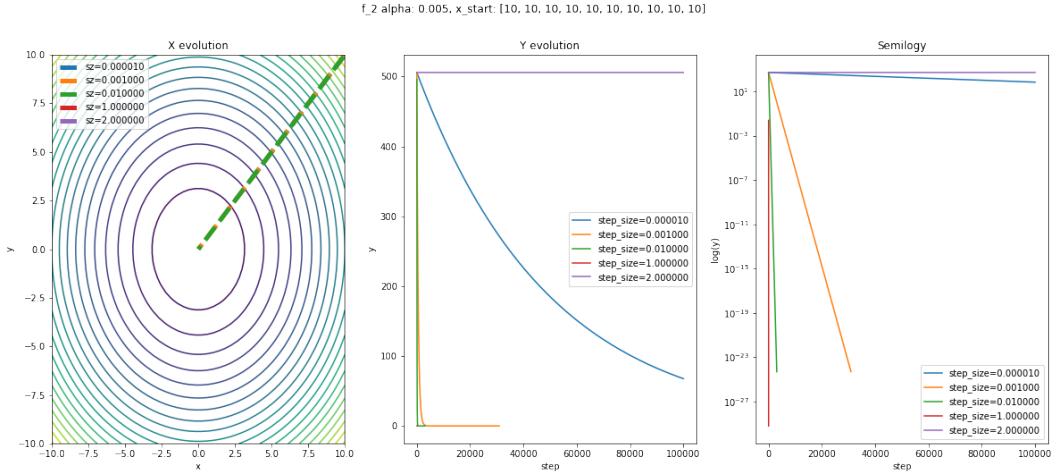


FIGURE 2.3

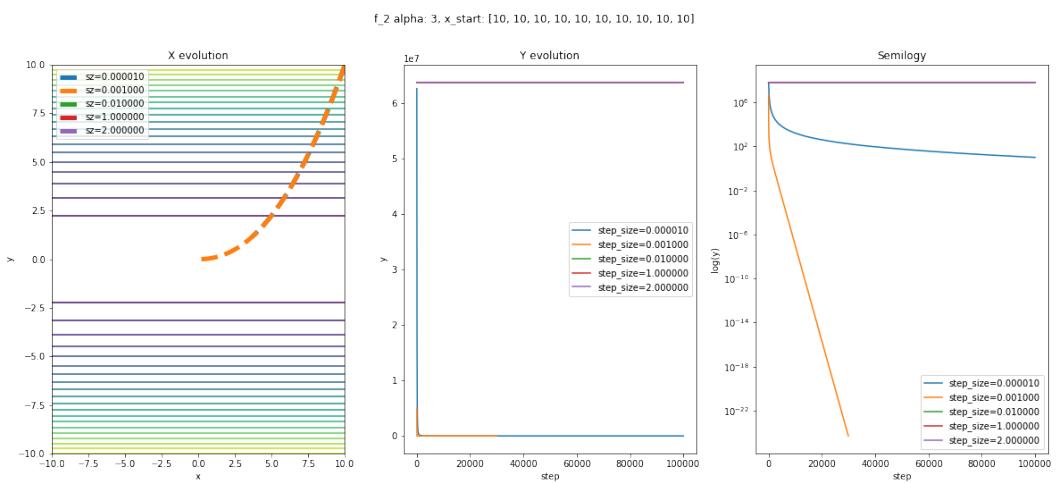


FIGURE 2.4

- La fonction $f_2^\alpha(x)$ reste assez simple (fonction convexe) ce qui permet de converger assez simplement.
- Même en complexifiant la fonction (ie : en prenant une grande valeur de α) on arrive à converger.
- Cependant on remarque que dans certains cas il nous faut plusieurs itérations.
- Cela peut être expliqué par des sauts trop grand ou trop petit à cause du *step_size*

2.3 Fonction 3

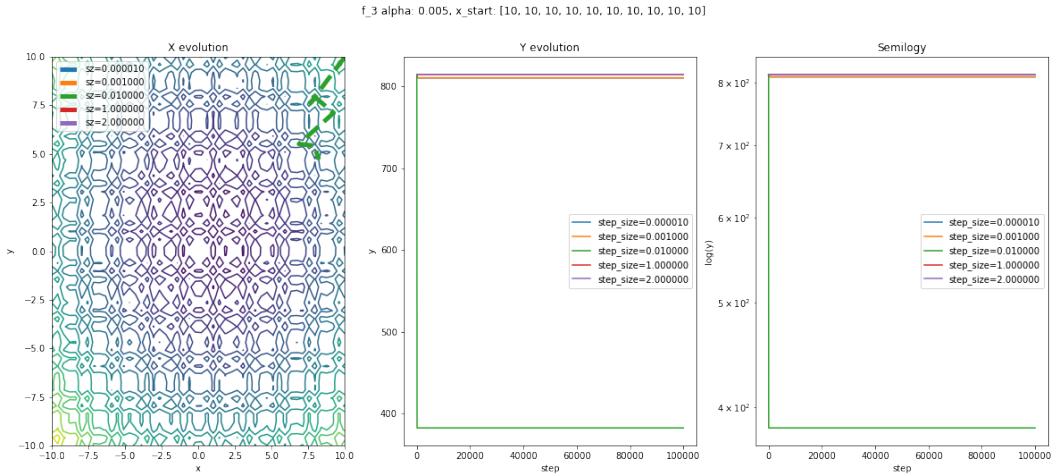


FIGURE 2.5

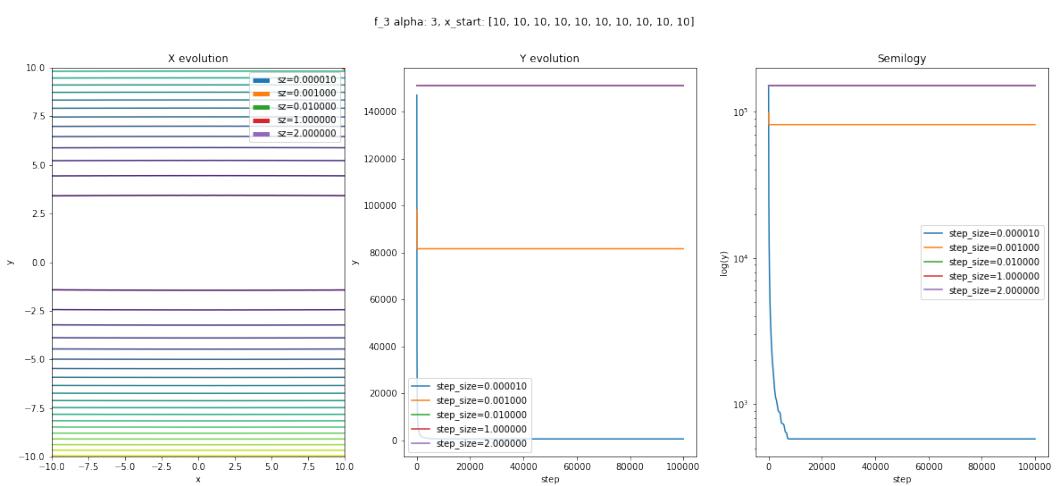


FIGURE 2.6

- La fonction $f_3^\alpha(x)$ est plus complexe que les deux dernières (admet plusieurs minimums locaux).
- En complexifiant la fonction (ie : en prenant une grande valeur de α) on ne converge pas.
- On remarque que l'algorithme peine à converger, on reste encore loin du minimum (0).
- On pourrait augmenter le nombre d'itération, mais cela va prendre beaucoup plus de temps.

2.4 Fonction 4

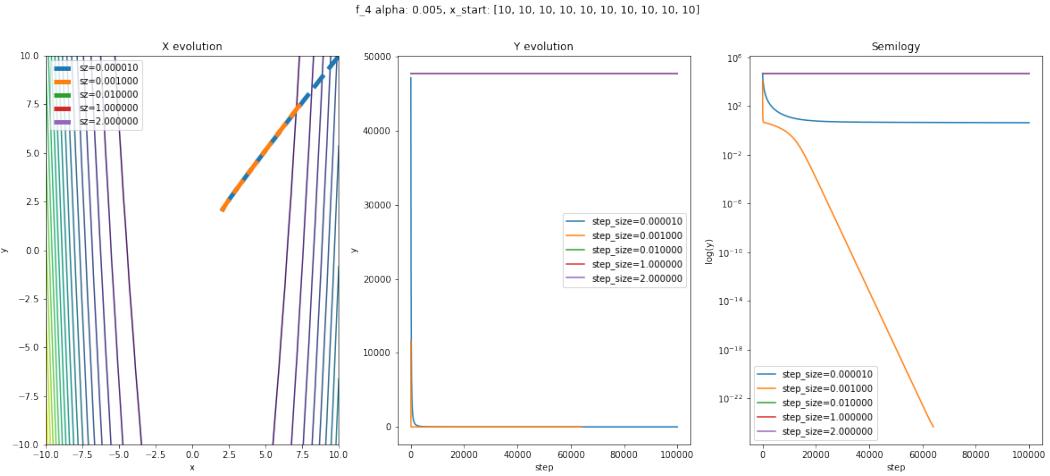


FIGURE 2.7

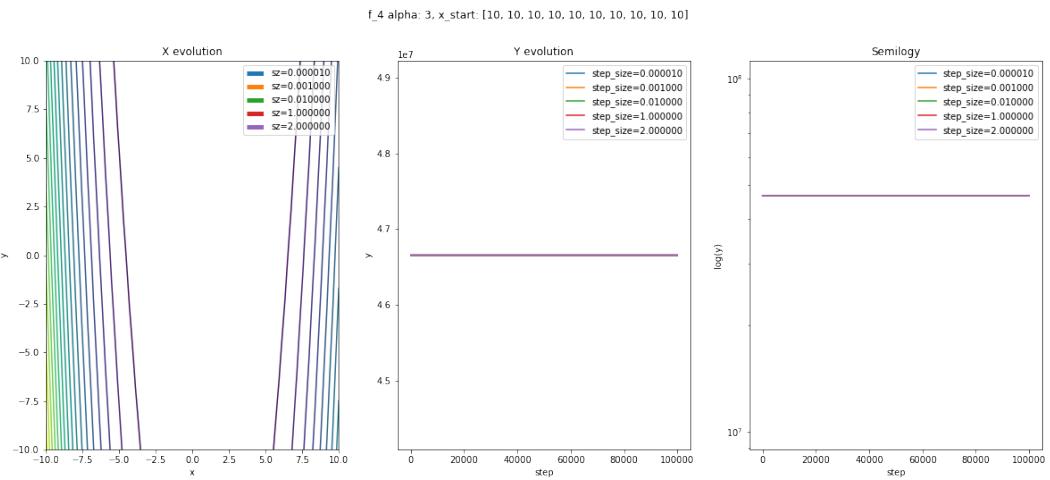


FIGURE 2.8

- La fonction $f_4^\alpha(x)$ est la plus complexe (admet plusieurs minimums locaux).
- En complexifiant la fonction (ie : en prenant une grande valeur de α) on ne converge pas.
- On remarque que l'algorithme peine à converger, on reste encore loin du minimum (0).
- On pourrait augmenter le nombre d'itération, mais cela va prendre beaucoup plus de temps.

3 Gradient Descent : Adaptatif Step Size

Dans cette partie nous allons automatiquement ajuster le `step_size` grâce à l'algorithme d'Armijo. De façon générale nous allons observer que l'algorithme converge plus rapidement. Par la suite nous allons observer l'évolution des X pendant la descente du gradient, en montrant les différentes valeurs du `step_size` ainsi que les valeurs logarithmique de Y .

3.1 Fonction 1

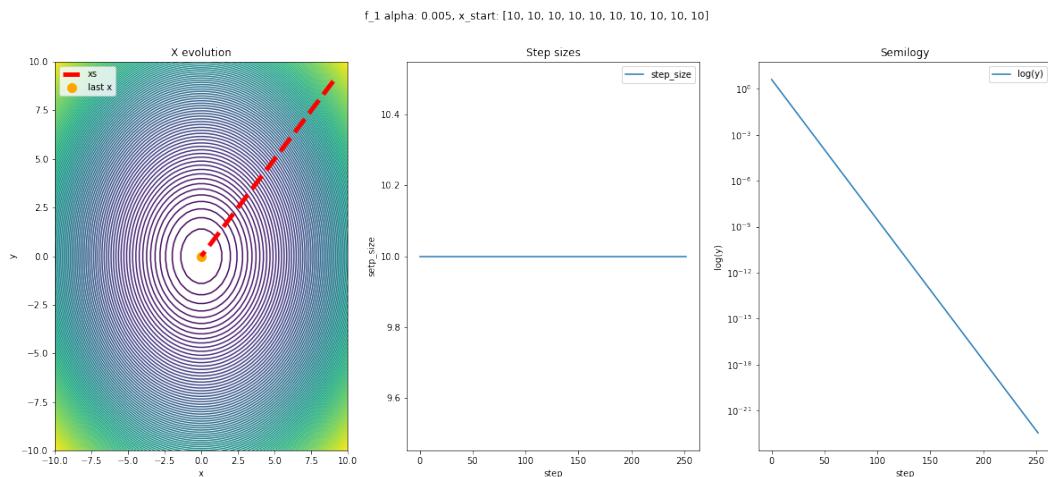


FIGURE 3.1

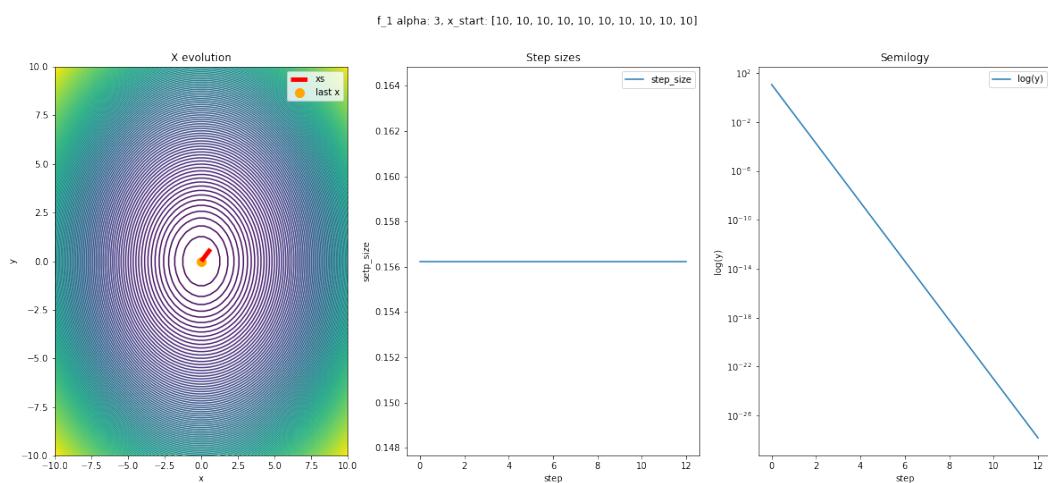


FIGURE 3.2

- On remarque que le nombre d'itération est beaucoup moins élevé.
- On converge beaucoup plus vite.

3.2 Fonction 2

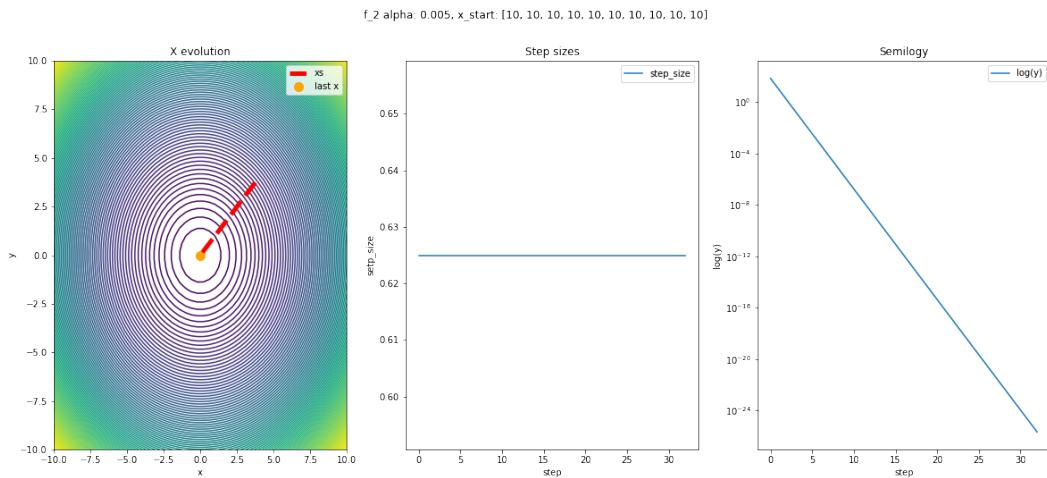


FIGURE 3.3

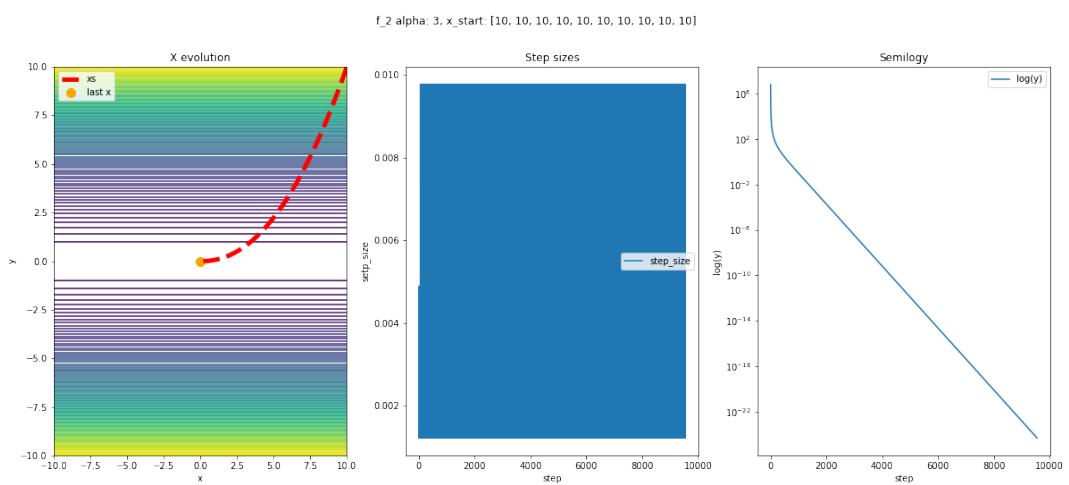


FIGURE 3.4

- On remarque que le nombre d'itération est beaucoup moins élevé.
- On converge beaucoup plus vite.
- On remarque aussi que le step *step_size* oscille entre de faible valeurs [0.02, 0.01].

3.3 Fonction 3

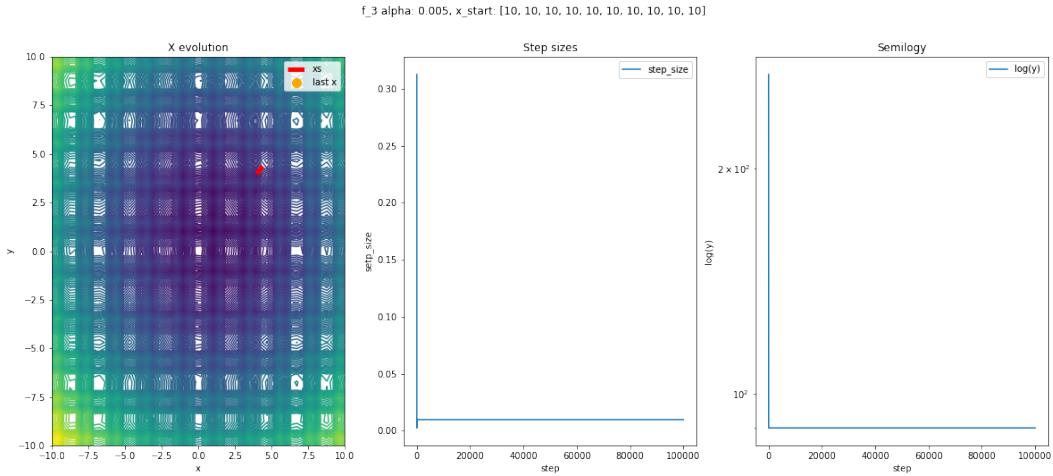


FIGURE 3.5

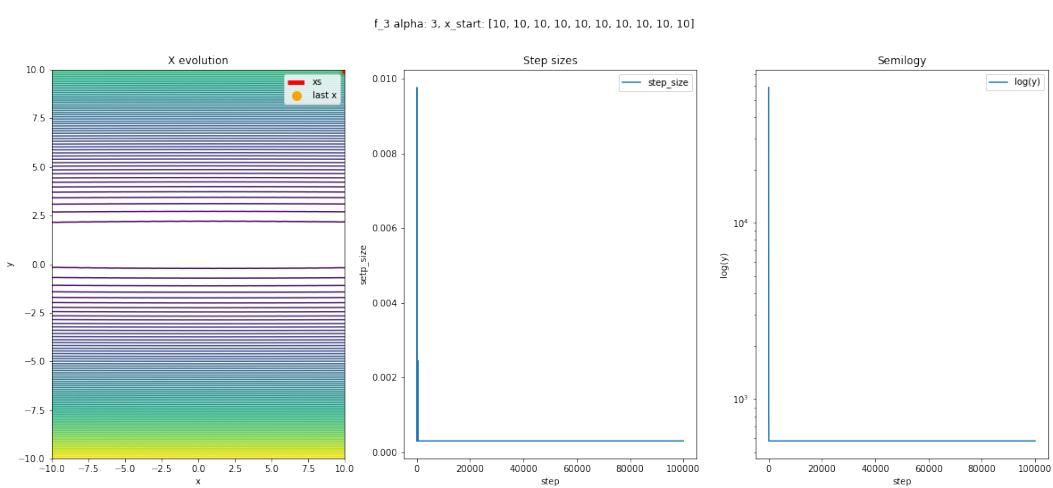


FIGURE 3.6

- La fonction $f_3^\alpha(x)$ est plus complexe que les deux dernières (admet plusieurs minimums locaux).
- On remarque que l'algorithme peine à converger, on reste encore loin du minimum (0), malgré un nombre élevé d'itération.
- On remarque aussi que le step $step_size$ oscille entre de faible valeurs [0.01, 0.02].

3.4 Fonction 4

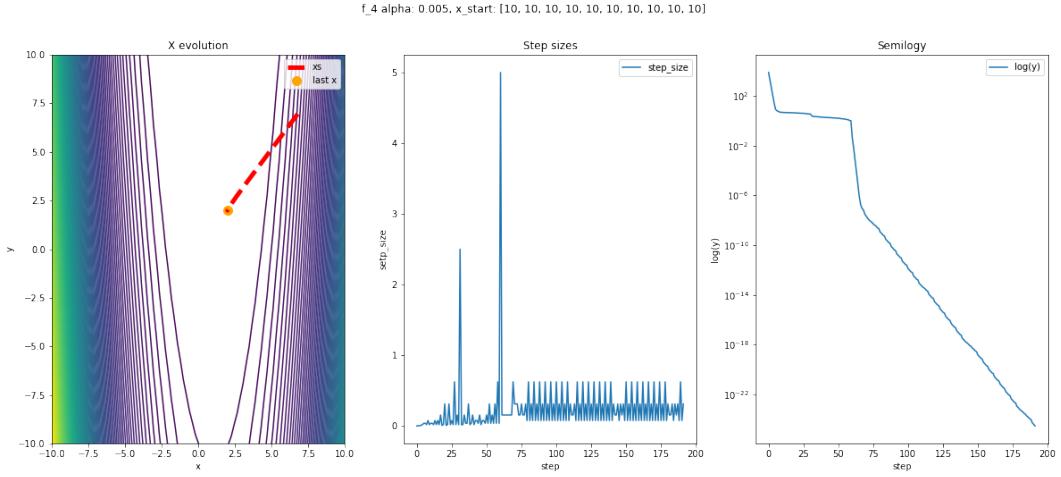


FIGURE 3.7

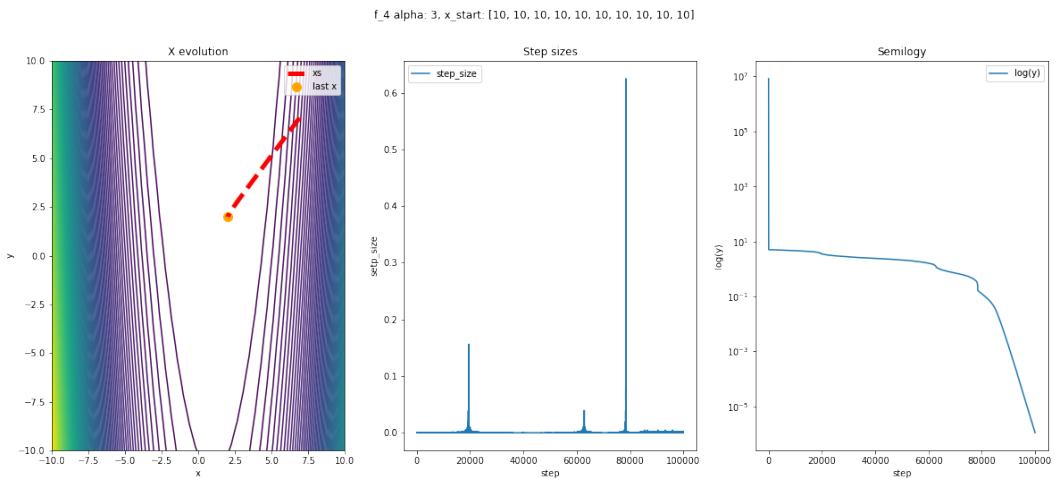


FIGURE 3.8

- La fonction $f_4^\alpha(x)$ est la plus complexe (admet plusieurs minimums locaux).
- On remarque que l'algorithme peine à converger, on reste encore loin du minimum (0).
- On remarque aussi que le step $step_size$ oscille entre des valeurs qui sont plus élevées que pour les fonctions précédentes [0.6, 0.01].

4 Gradient Descent : quasi-Newton Methode

Dans cette partie nous allons observer une descente de gradient avec la méthode quasi-Newton. Pour cela nous devons calculer les dérivées seconde de chaque fonction :

$$f''_1^\alpha(x) = 2.n.\alpha \quad (4.1)$$

$$f_2''(x) = n \cdot 10^{\alpha \frac{i-1}{n-1}} \quad (4.2)$$

$$f_3''(x) = \sum_{i=1}^n (2.10^{\alpha \frac{i-1}{n-1}} + 10.4\pi^2 \cos(2\pi(x_i - 1))) \quad (4.3)$$

$$f_4''(x) = \begin{cases} \sum_{i=1}^n 4.10^{\alpha} \cdot (xi - 1) \\ \sum_{i=1}^n -4.10^{\alpha} 2 \cdot (xi - 1) \end{cases} \quad (4.4)$$

4.1 Fonction 1

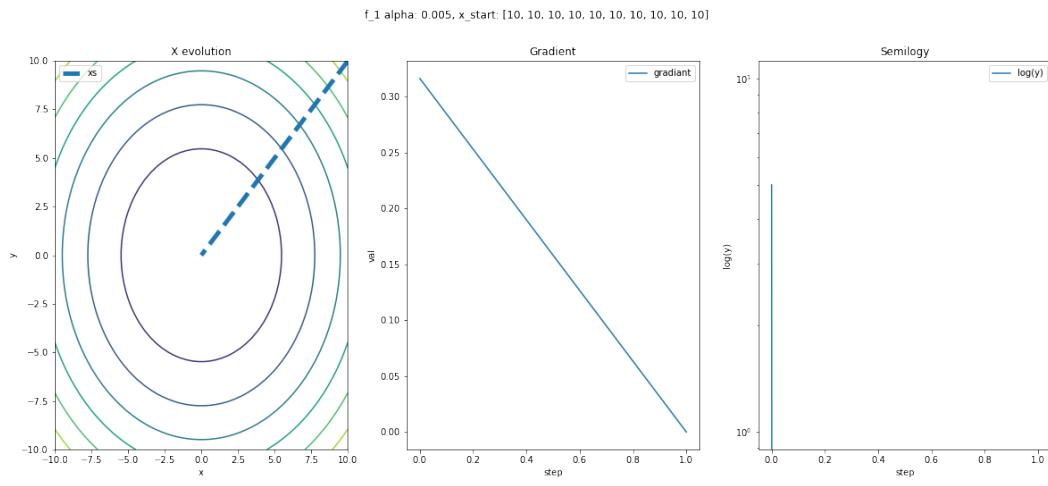


FIGURE 4.1

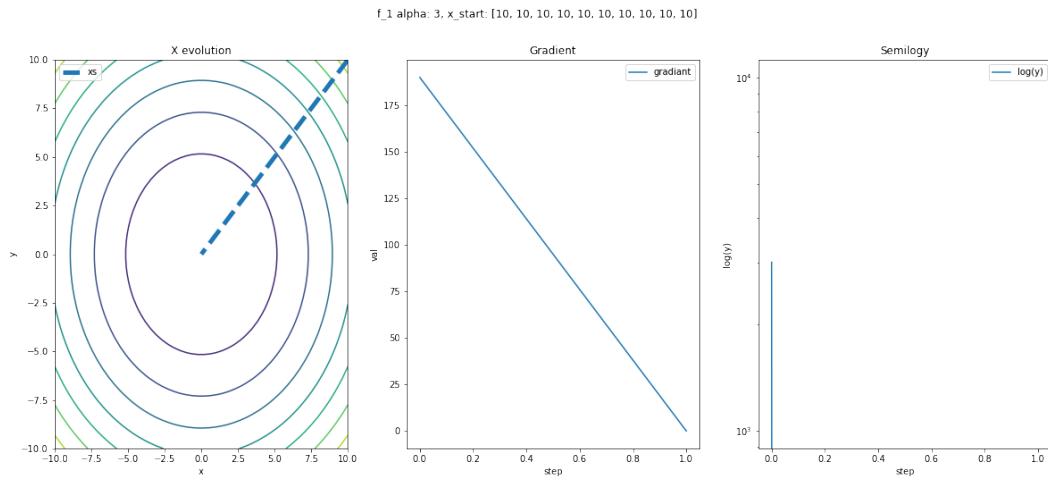


FIGURE 4.2

- La fonction $f_1^\alpha(x)$ reste assez simple (fonction convexe) ce qui permet de converger assez simplement.
- On peut remarquer ici que le facteur α impacte énormément sur la descente du gradient.
- Au plus α est grand, au plus le *step_size* doit être petit.

4.2 Fonction 2

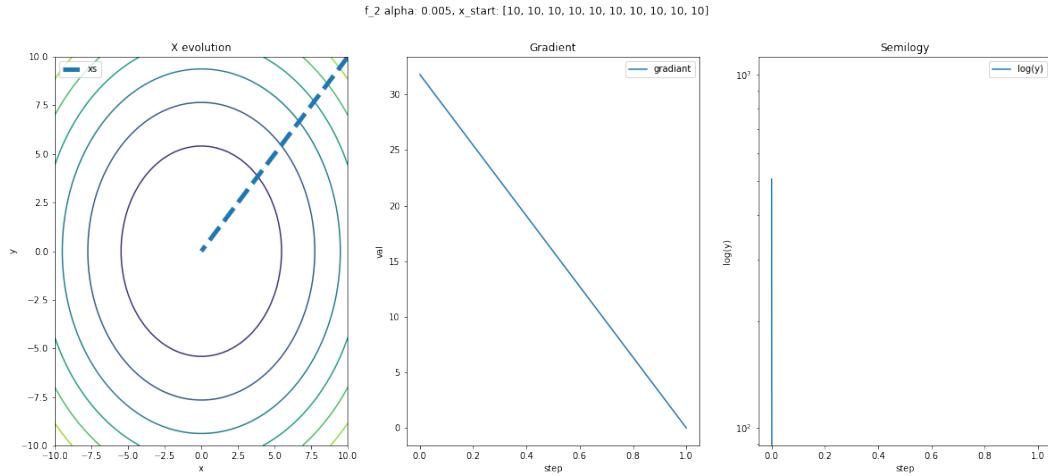


FIGURE 4.3

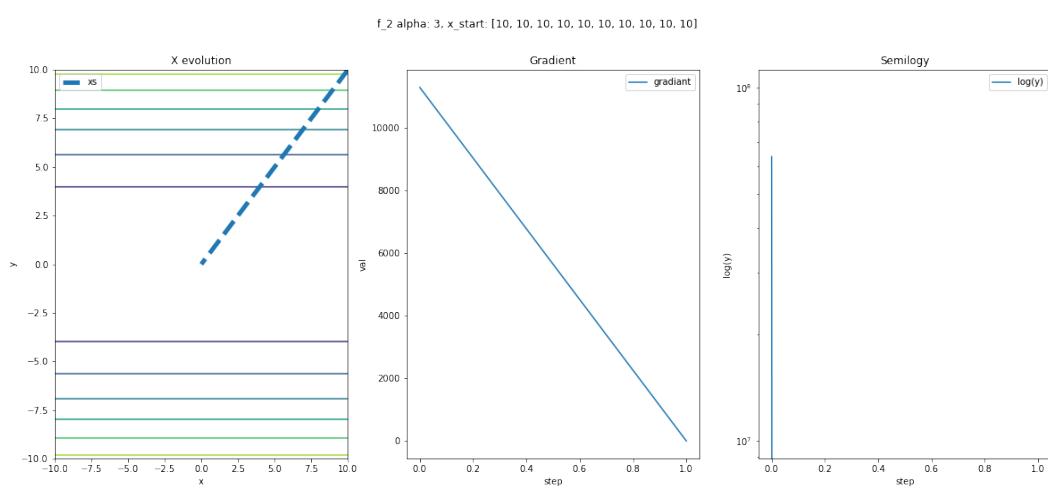


FIGURE 4.4

- La fonction $f_2^\alpha(x)$ reste assez simple (fonction convexe) ce qui permet de converger assez simplement.
- On peut remarquer ici que le facteur α impacte énormément la fonction et par suite impacte sur la descente du gradient.
- Au plus α est grand, au plus le *step_size* doit être petit.

4.3 Fonction 3

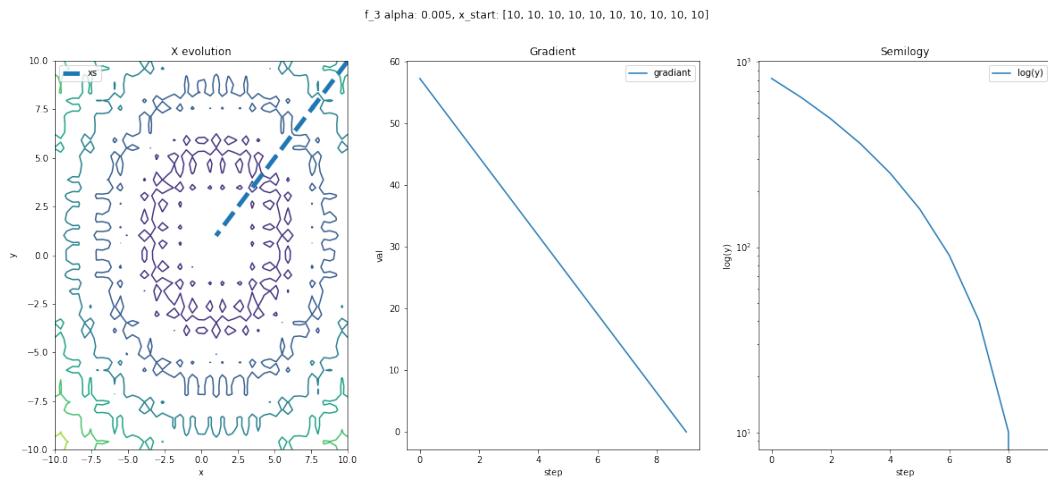


FIGURE 4.5

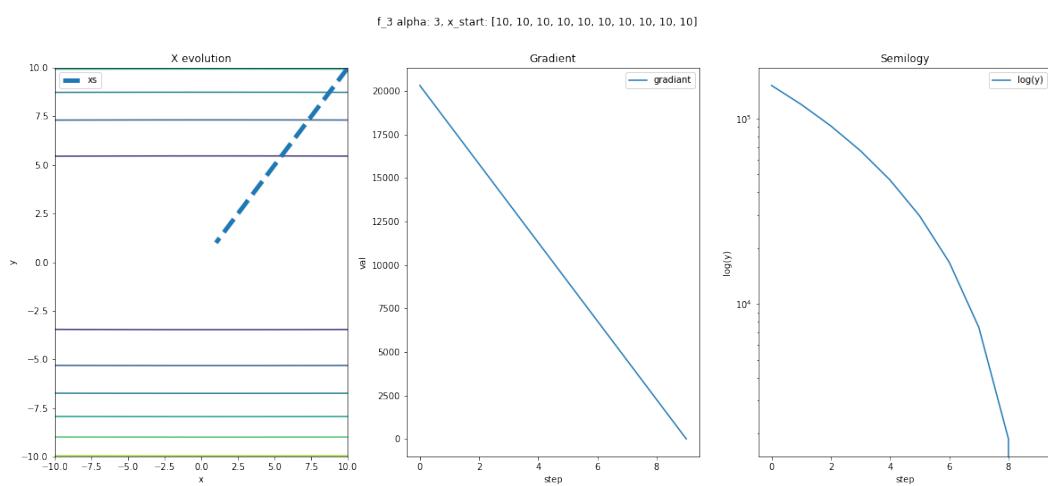


FIGURE 4.6

- La fonction $f_3^\alpha(x)$ est plus complexe que les deux dernières (admet plusieurs minimums locaux).
- On remarque que l'algorithme peine à converger, on reste encore loin du minimum (0).
- On peut remarquer ici que le facteur α impacte énormément la fonction et par suite impacte sur la descente du gradient.

4.4 Fonction 4

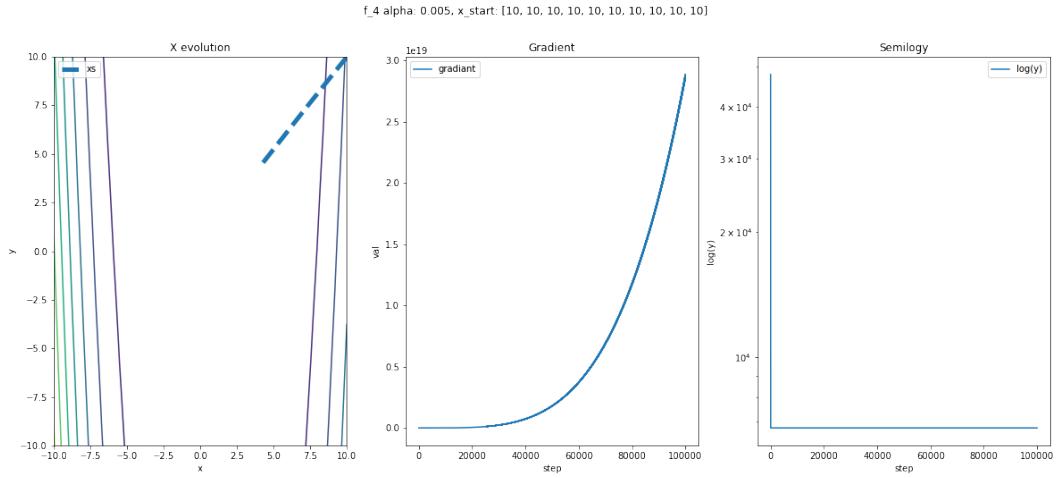


FIGURE 4.7

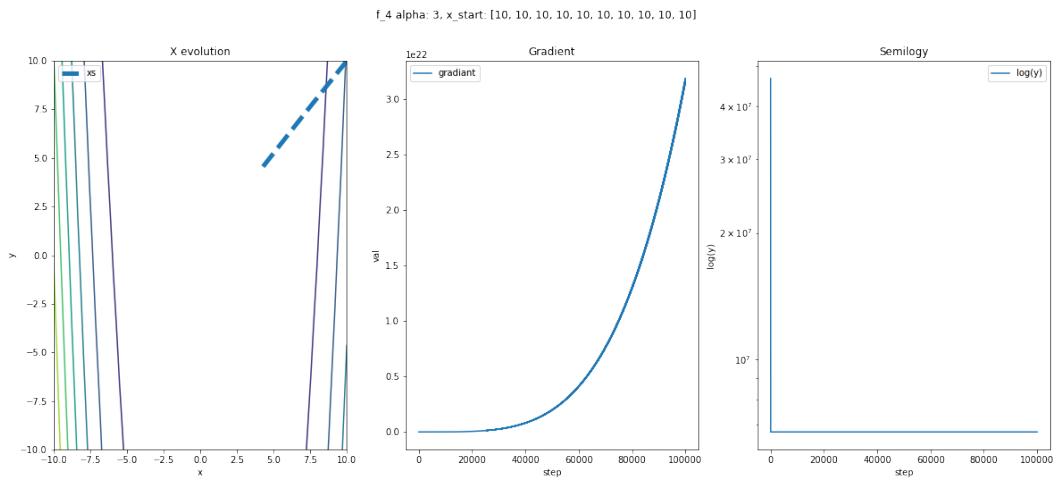


FIGURE 4.8

- La fonction $f_4^\alpha(x)$ est la plus complexe (admet plusieurs minimums locaux).
- On remarque que l'algorithme peine à converger, on reste encore loin du minimum (0).

5 Comparaison Newton VS BFGS

Dans cette partie nous comparer la méthode Newton à la méthode BFGS de la bibliothèque scipy.

5.1 Fonction 1

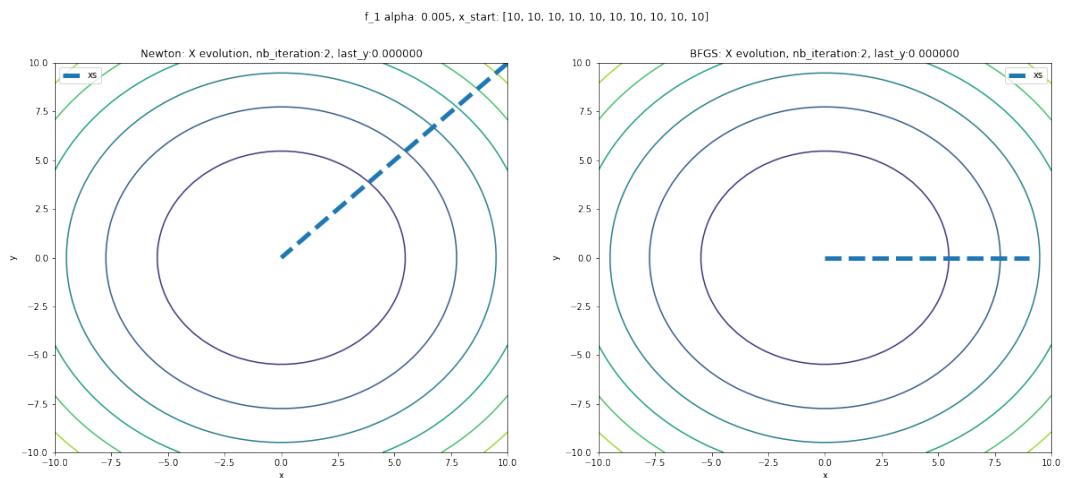


FIGURE 5.1

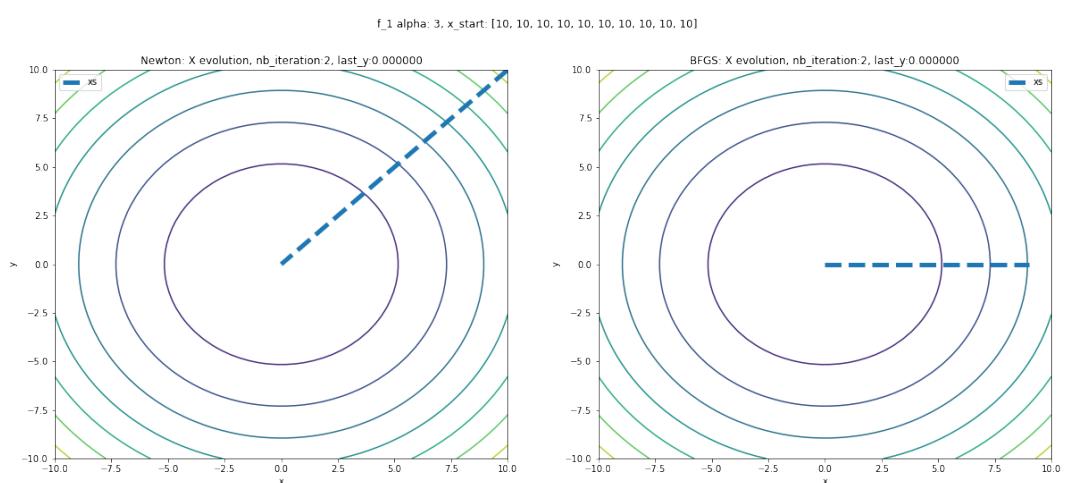


FIGURE 5.2

— On peut remarquer ici que les deux méthodes sont équivalentes.

5.2 Fonction 2

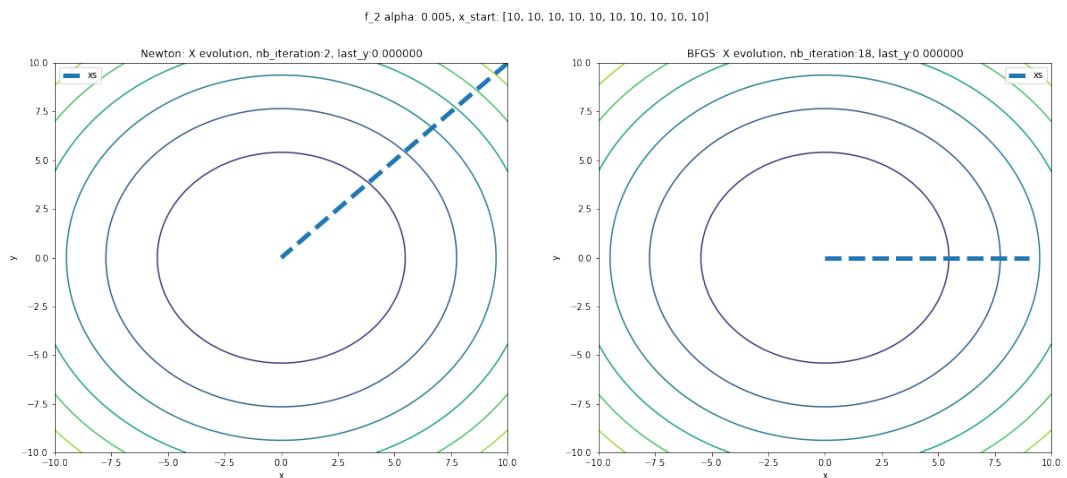


FIGURE 5.3

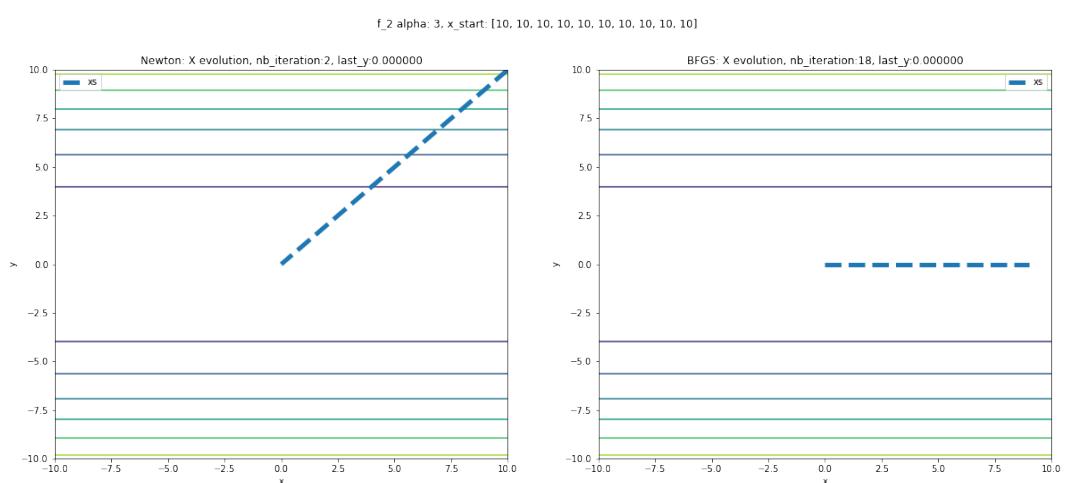


FIGURE 5.4

— On peut remarquer ici que les deux méthodes sont équivalentes.

5.3 Fonction 3

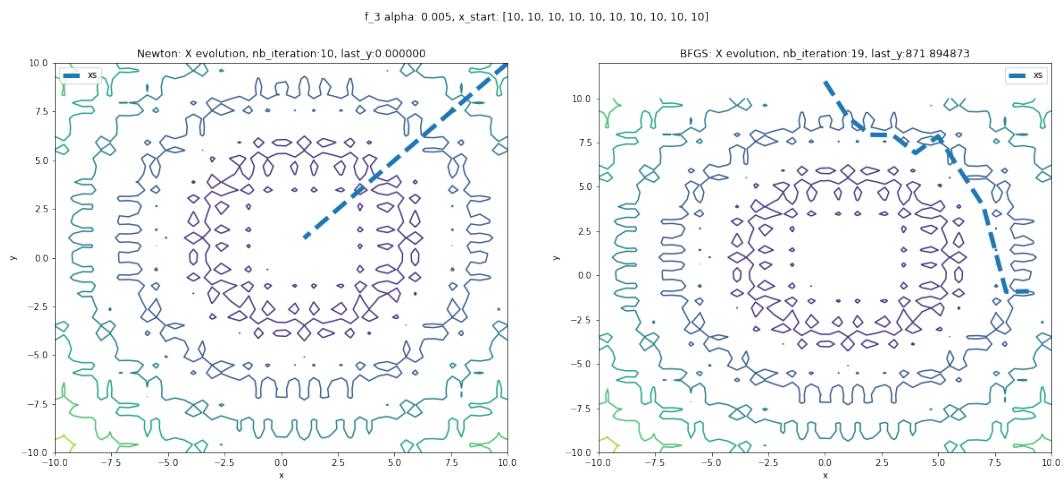


FIGURE 5.5

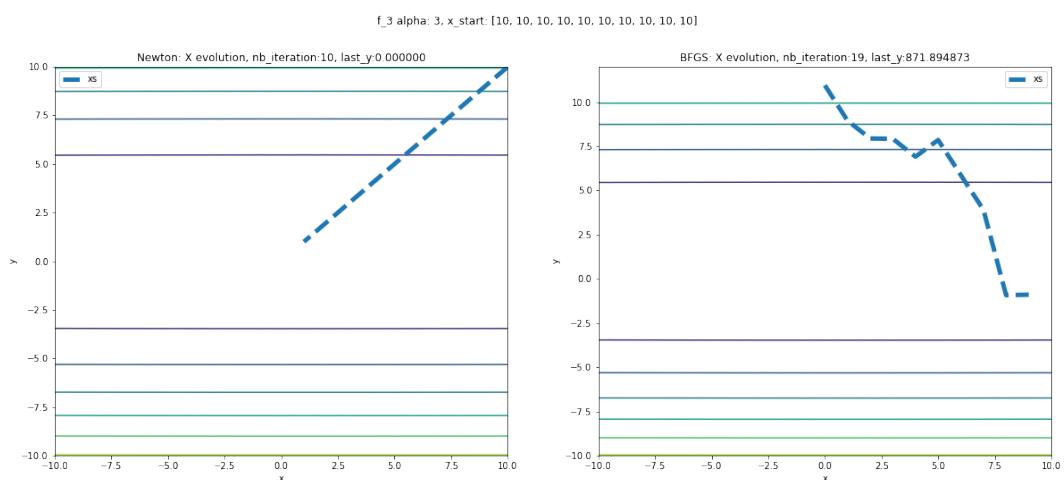


FIGURE 5.6

— On peut remarquer ici que la méthode Newton est meilleur que BFGS.

5.4 Fonction 4

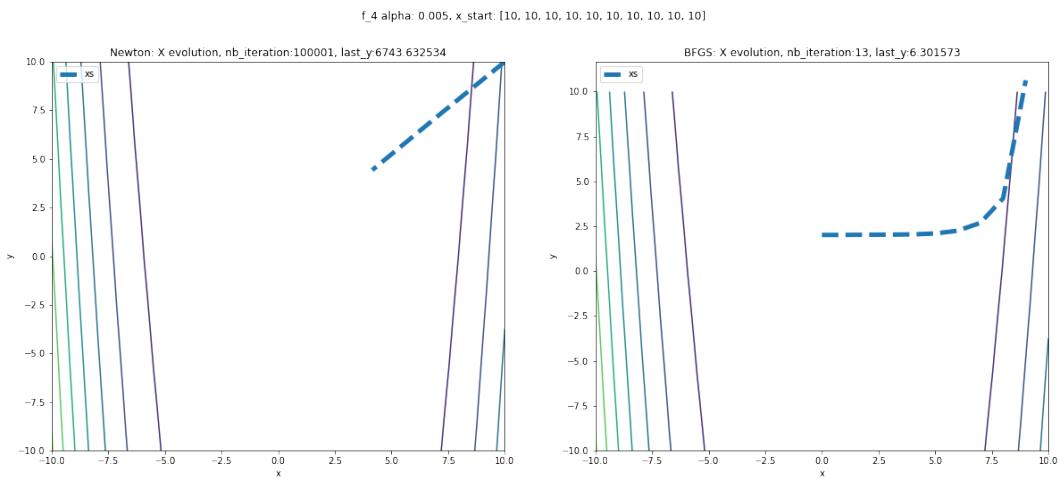


FIGURE 5.7

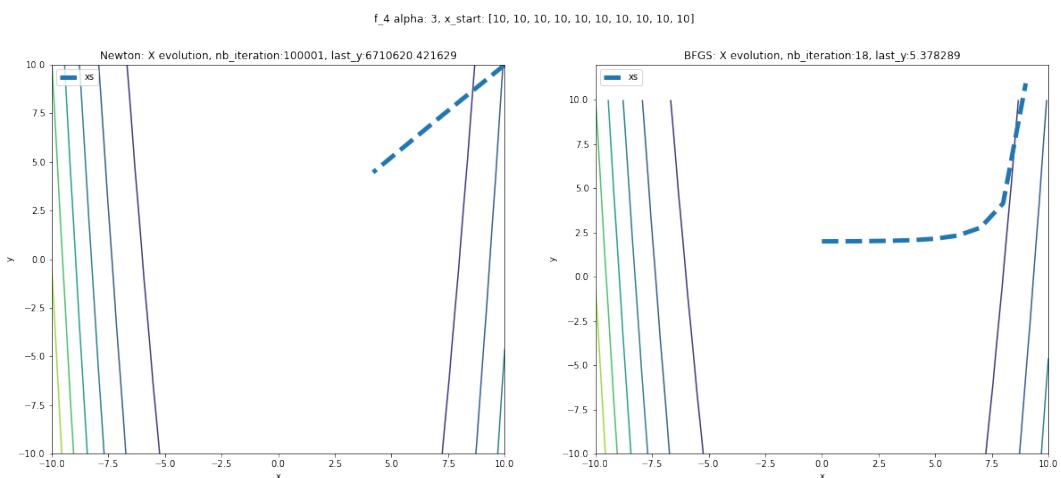


FIGURE 5.8

— On peut remarquer ici que la méthode BFGS est meilleur que Newton.