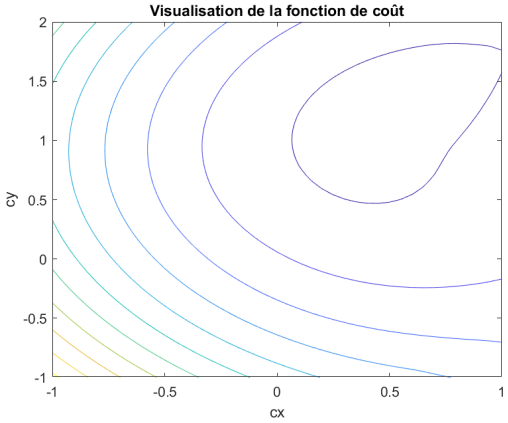
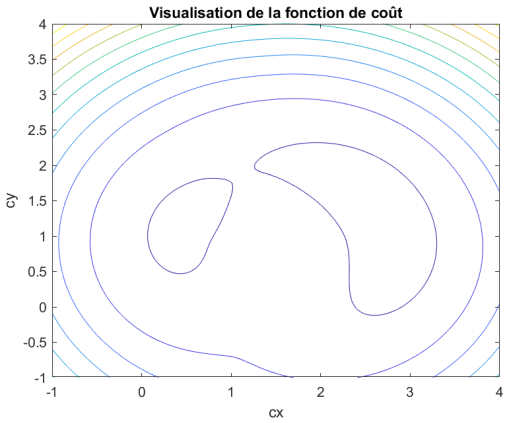
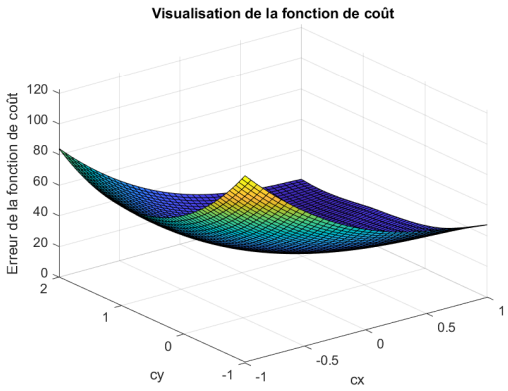
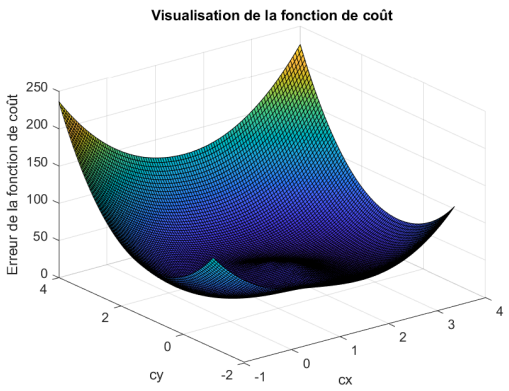
Compte-rendu - Projet d'optimisation continue

Estimation robuste du centre d'une pièce circulaire

Justin Bossard Tom Mafille Ruben Verchere

Octobre et novembre 2024

1. On obtient les tracés suivants, avec un pas arbitraire de $0,05$, pour la fonction de coût $\mathcal{C}_{TLS}(c_x, c_y) = \sum_{i=1}^{n}(D_i - R)^2$:

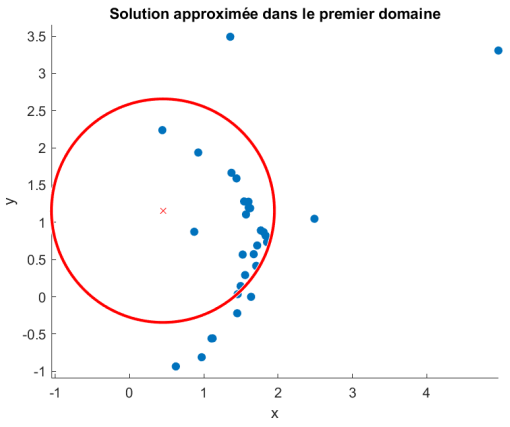
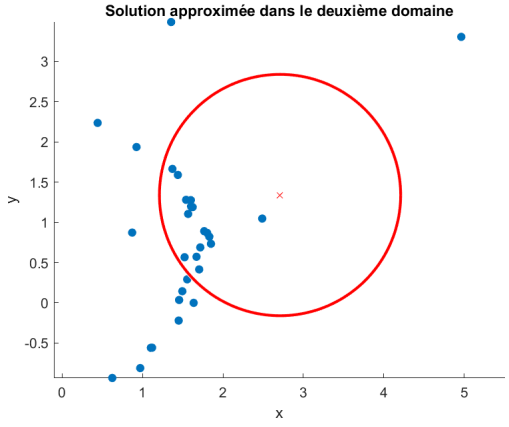
Sur $[-1;1] \times [-1;2]$	Sur $[-1;4] \times [-1;4]$
	
	

Sur la première figure, on a qu'un seul minimum, tandis que sur la seconde on en observe deux, qu'il faut départager. Il est donc préférable d'avoir une plus grande échelle, permettant de visualiser tous les minimums, plutôt qu'une petite fenêtre éliminant d'office des solutions potentielles.

2. Pour déterminer c_x et c_y à 10^{-4} près, on estime $N = \frac{(longueur\ intervalle\ x)\times(longueur\ intervalle\ y)}{pas^2}$ fois la fonction de coût $\mathcal{C}_{TLS}$, dans la mesure on l'on passe dans deux boucles `for`, sur l'intervalle `départ:pas:arrivée`.

   On a donc, sur $[-1;1]\times[-1;2]$, $N_1 = \frac{2\times3}{10^{-8}} = 6\times10^8$ estimations de la fonction de coût, et $N_2 = 25\times10^8$ estimations pour l'intervalle $[-1;4]\times[-1;4]$.

   On obtient les cercles et les nuages de points suivants :

Sur $[-1;1]\times[-1;2]$	Sur $[-1;4]\times[-1;4]$
	

   On observe que les points aberrants influencent grandement le centre obtenu : en effet, on peut tomber dans un minimum local qui n'est pas le minimum global. La méthode n'est donc pas adaptée pour pouvoir déterminer le centre.

   Pour pouvoir approximer le rayon R, il faut prendre en compte le nombre d'estimations du rayon par boucle `for`, qui est donc de $N_R = \frac{2{,}5-0{,}5}{10^{-4}} = 2\times10^{-4}$.

   Le nombre d'estimations de $\mathcal{C}_{TLS}$ est dorénavant de $N_1' = N_1\times N_R = 1{,}2\times10^{13}$ sur $[-1;1]\times[-1;2]$, et de $N_2' = N_2\times N_R = 5\times10^{13}$ sur $[-1;4]\times[-1;4]$.

3. En calculant le gradient à la main :

$$\nabla\mathcal{C}_{TLS}(c_x,c_y) = \begin{pmatrix} \frac{\partial}{\partial x}\sum_{i=1}^{n}(D_i-R)^2 \\ \frac{\partial}{\partial y}\sum_{i=1}^{n}(D_i-R)^2 \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{i=1}^{n}\frac{\partial}{\partial x_i}(D_i-R)^2 \\ \sum_{i=1}^{n}\frac{\partial}{\partial y_i}(D_i-R)^2 \end{pmatrix}$$

$$\nabla\mathcal{C}_{TLS}(c_x,c_y) = \begin{pmatrix} 2\sum_{i=1}^{n}(c_x-x_i)\left(1-\frac{R}{D_i}\right) \\ 2\sum_{i=1}^{n}(c_y-y_i)\left(1-\frac{R}{D_i}\right) \end{pmatrix}$$

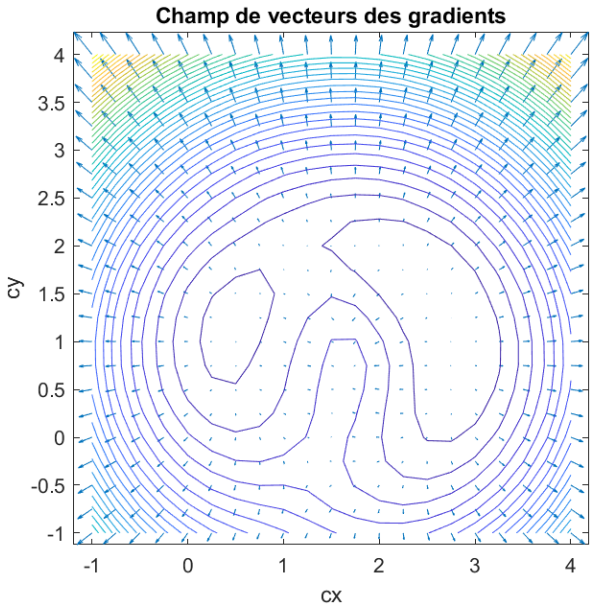
4. On choisit arbitrairement, pour tester le gradient, les points $(0;0)$, $(1;1)$, $(2,5;-0,5)$, $(3,5;2)$, $(0;3)$. On calcule le taux d'accroissement avec Matlab dans la fonction `gradient_diff_finie_1(point, delta)`, et on obtient les résultats suivants :

x	y	$\frac{\partial}{\partial x}\mathcal{C}_{TLS}$	t_x	$\frac{\partial}{\partial y}\mathcal{C}_{TLS}$	t_y	η_x	η_y
0	0	-21.4640	-21.4617	-21.0858	-21.0846	0.0001	0.0001
1	1	14.2961	14.2968	-6.0875	-6.0876	0	0
2.5	-0.5	3.2742	3.2758	-20.7998	-20.7982	0.0005	0.0001
3.5	2	46.7235	46.7259	30.1031	30.1048	0.0001	0.0001
0	3	-42.8043	-42.8024	61.9556	61.9579	0	0

En notant $t_x = \frac{\mathcal{C}_{TLS}(x+\Delta_x,\, y)-\mathcal{C}_{TLS}(x,\, y)}{\Delta_x}$ et $t_y = \frac{\mathcal{C}_{TLS}(x,\, y+\Delta_y)-\mathcal{C}_{TLS}(x,\, y)}{\Delta_y}$.
On note les erreurs relatives $\eta_x = \frac{|\frac{\partial}{\partial x}\mathcal{C}_{TLS}-t_x|}{|\frac{\partial}{\partial x}\mathcal{C}_{TLS}|}$ et $\eta_y = \frac{|\frac{\partial}{\partial y}\mathcal{C}_{TLS}-t_y|}{|\frac{\partial}{\partial y}\mathcal{C}_{TLS}|}$.

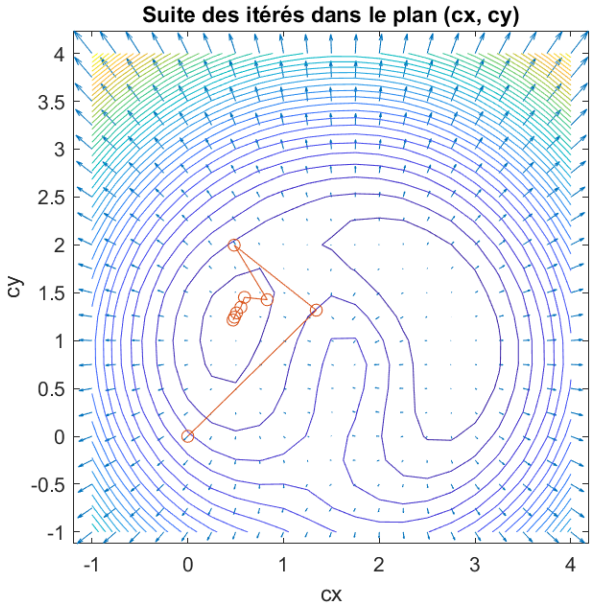
On a choisit $(\Delta_x;\Delta_y) = (10^{-4};10^{-4})$, ce qui est suffisament précis, puisqu'on voit que l'erreur relative est sytématiquement négligeable.

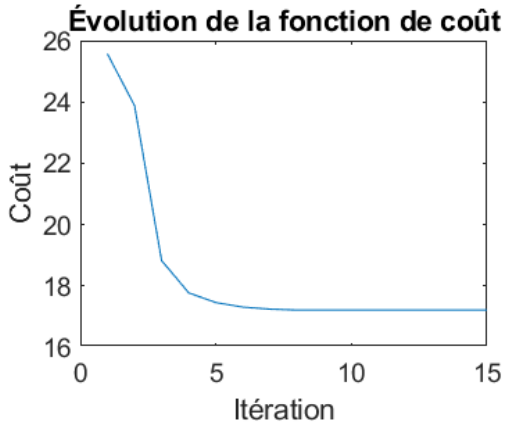
5. En représentant le champ de gradient avec la fonction `quiver`, avec les lignes de contours, avec un pas d'échantillonnage de $0,25$, on a :

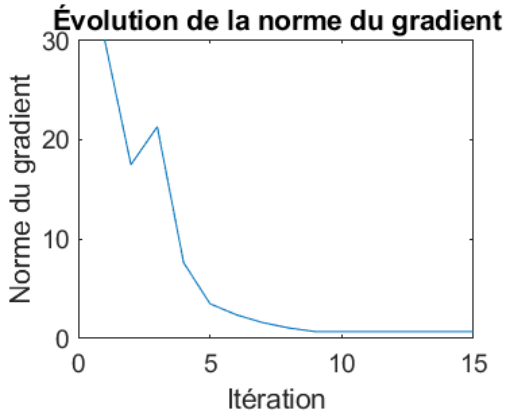


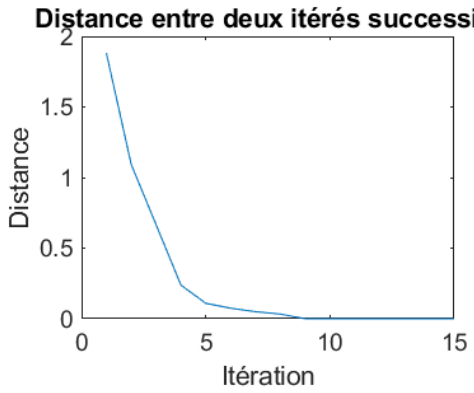
Le gradient est bien orthogonal aux lignes de niveaux.

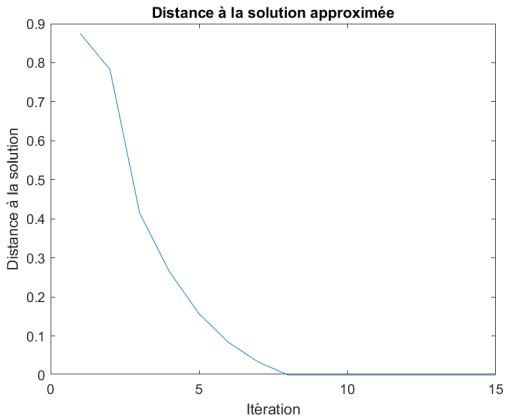
6. Par la méthode des plus fortes pentes, avec l'algorithme de Fletcher et Lemaréchal, on obtient la solution suivante :



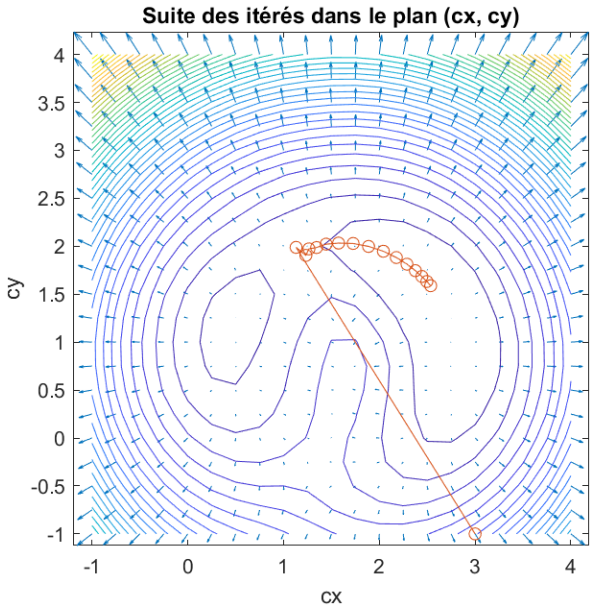


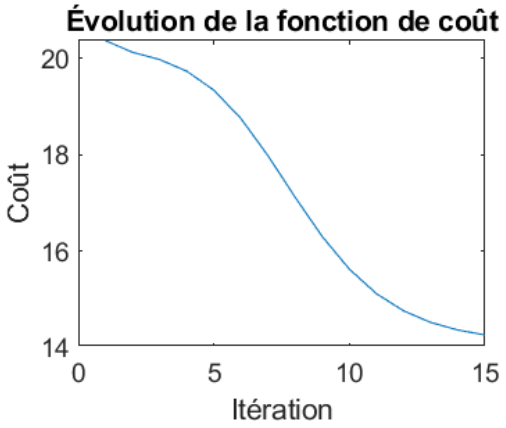


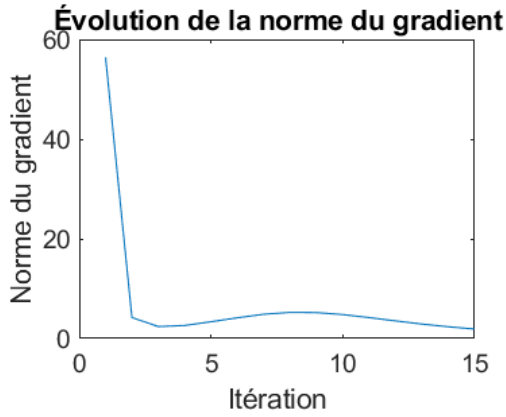


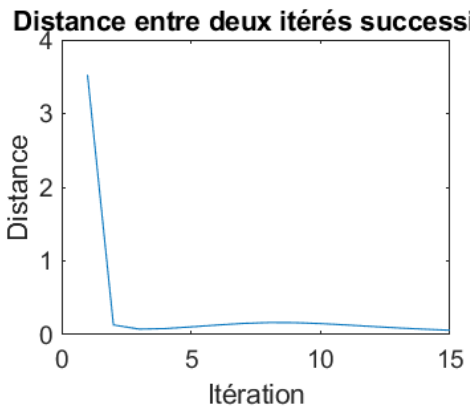


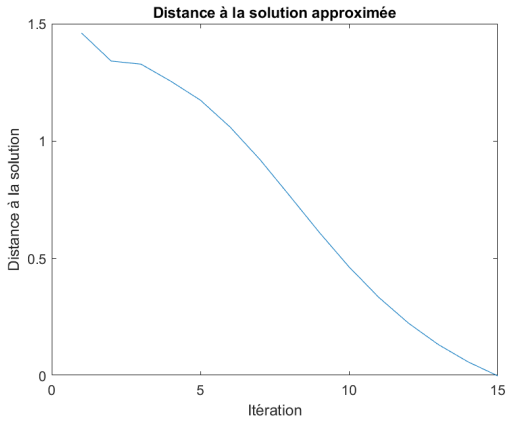
7. Selon le point de départ, on peut tomber dans le mauvais minimum, et donc avoir une solution erronée, comme l'illustre la figure suivante :





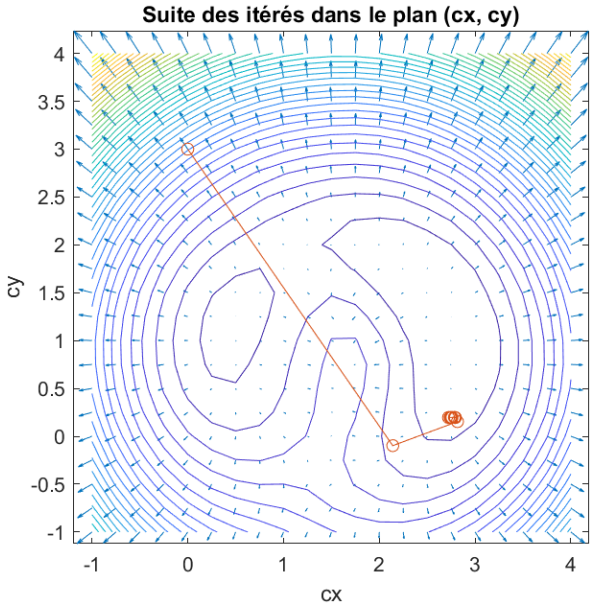


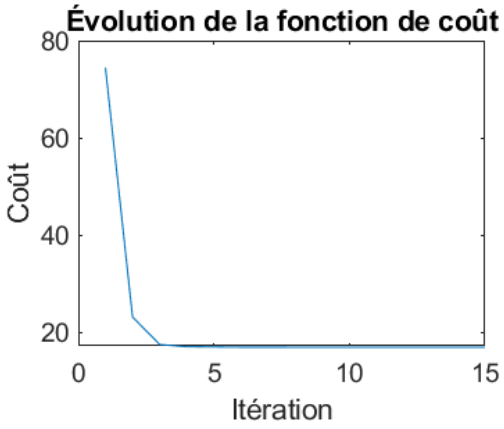


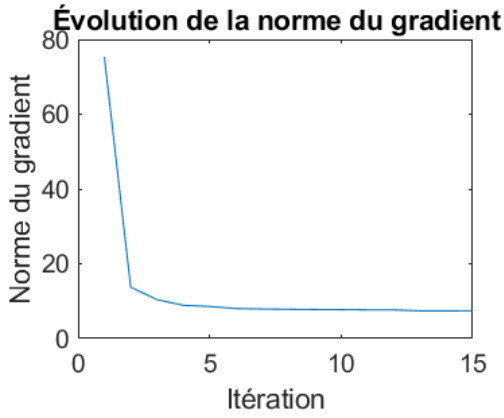


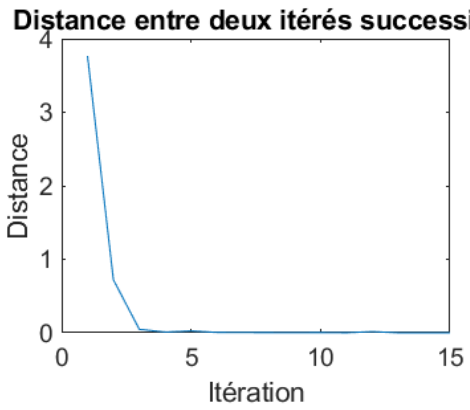
Ainsi, il faut pour utiliser cet algorithme de manière pertinente avoir au préalable une idée de la solution voulue.

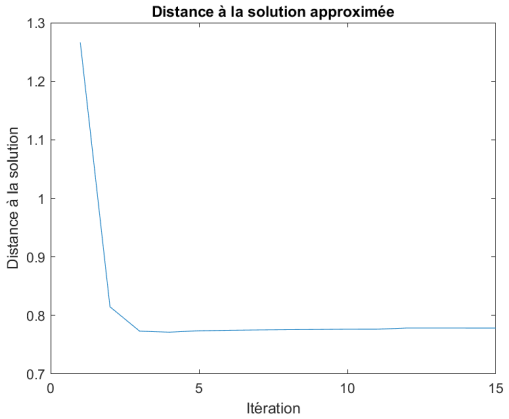
8. Avec la méthode de quasi-Newton, on obtient la convergence suivante :





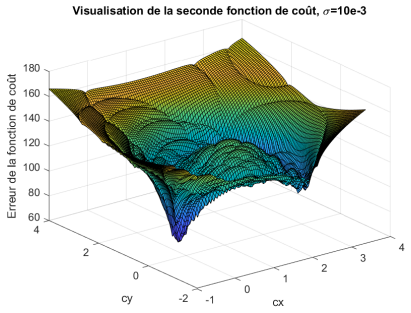
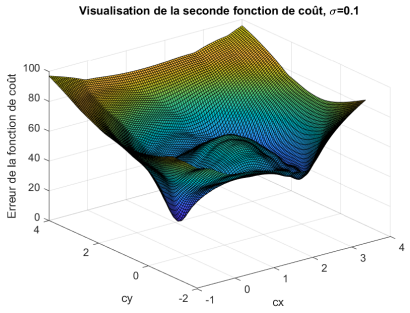
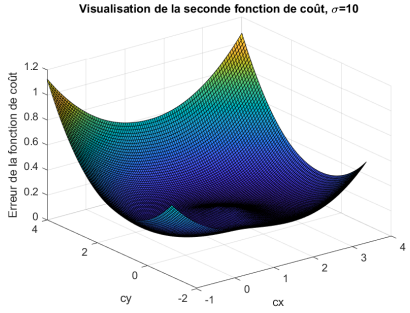
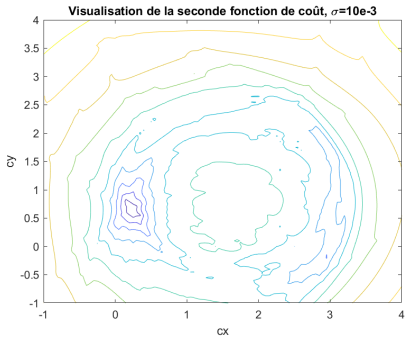
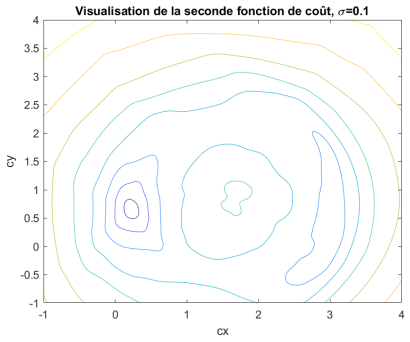
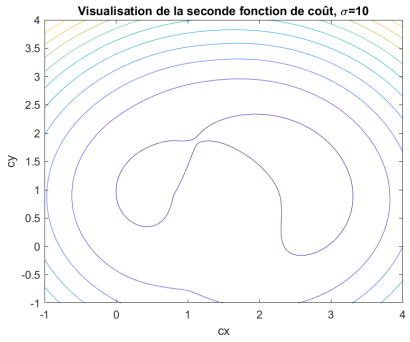
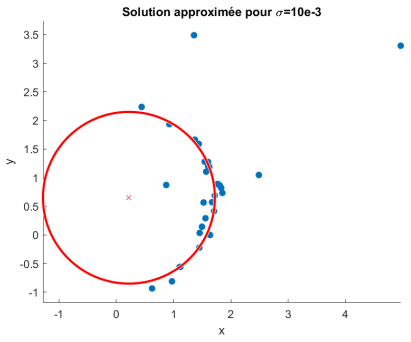
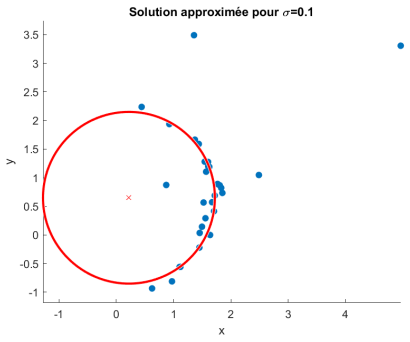
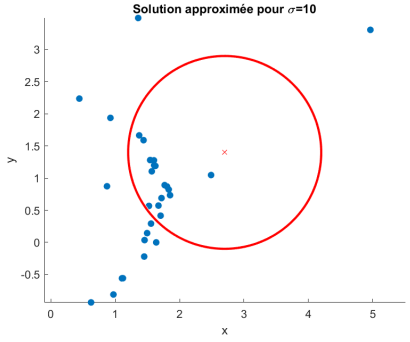






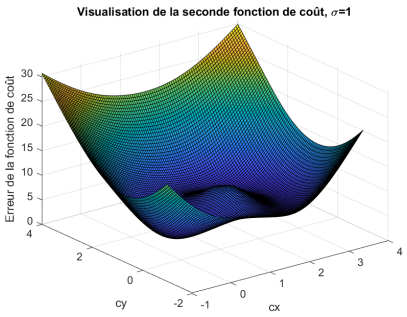
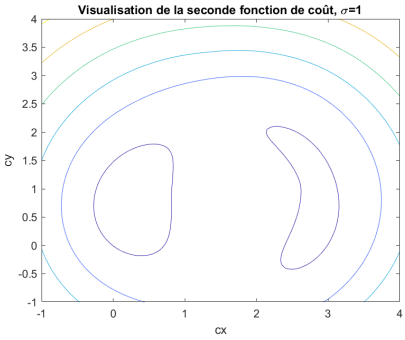
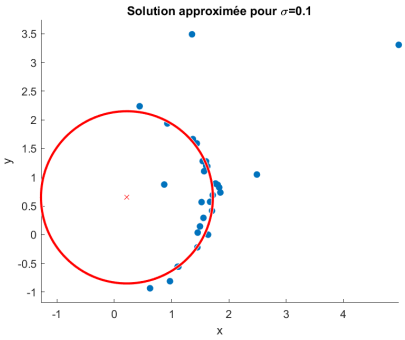
Nous convergons vers le mauvais minimum, on observe toutefois une convergence plus rapide qu'avec l'algorithme de Fletcher et Lemaréchal. La distance à la solution est cependant très importante par rapport au susdit algorithme.

9. On considère la nouvelle fonction de coût $\mathcal{C}'_{TLS}(c_x, c_y, \sigma) = \frac{1}{2}\sum_{i=1}^{n} \log\left(1 + \frac{(D_i - R)^2}{\sigma^2}\right)$, on obtient les résultats suivants :

$\sigma = 10^{-3}$	$\sigma = 0.1$	$\sigma = 10$
		
		
		

On peut voir que pour une valeur de σ trop faible, la précision sera très grande, mais que les variations sont exacerbées : cela peut avoir un impact sur la cohérence des résultats, et faire apparaître des anomalies ponctuelles tels que des *extrema* ne représentant pas la réalité. De plus, le temps de calcul est grand et le programme perd donc en efficacité. A l'inverse, pour une valeur de σ trop grande, on aura un lissage trop prononcé qui omettra les nuances, bien que l'efficacité soit grande. Il est donc nécessaire de privilégier une valeur de σ qui soit un bon compromis entre ces facteurs : une valeur située entre 0.1 et 10 telle que 1 semble être ce compromis recherché.

Pour $\sigma = 1$, on a :

Fonction de coût	Contour	Solution approchée
		

10. 1. Pour la nouvelle expression du gradient, on a :

$$\nabla \mathcal{C}'_{TLS}(c_x, c_y) = \begin{pmatrix} \frac{\partial}{\partial x} \frac{1}{2} \sum_{i=1}^{n} \log\left(1 + \frac{(D_i - R)^2}{\sigma^2}\right) \\ \frac{\partial}{\partial y} \frac{1}{2} \sum_{i=1}^{n} \log\left(1 + \frac{(D_i - R)^2}{\sigma^2}\right) \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{2} \sum_{i=1}^{n} \frac{1}{1 + \frac{(D_i - R)^2}{\sigma^2}} \frac{\partial}{\partial x_i} \left(\frac{(D_i - R)^2}{\sigma^2}\right) \\ \frac{1}{2} \sum_{i=1}^{n} \frac{1}{1 + \frac{(D_i - R)^2}{\sigma^2}} \frac{\partial}{\partial y_i} \left(\frac{(D_i - R)^2}{\sigma^2}\right) \end{pmatrix}$$

$$\nabla \mathcal{C}'_{TLS}(c_x, c_y) = \begin{pmatrix} \sum_{i=1}^{n} \frac{D_i - R}{1 + \frac{(D_i - R)^2}{\sigma^2}} \frac{c_x - x_i}{D_i \sigma^2} \\ \sum_{i=1}^{n} \frac{D_i - R}{1 + \frac{(D_i - R)^2}{\sigma^2}} \frac{c_y - y_i}{D_i \sigma^2} \end{pmatrix}$$

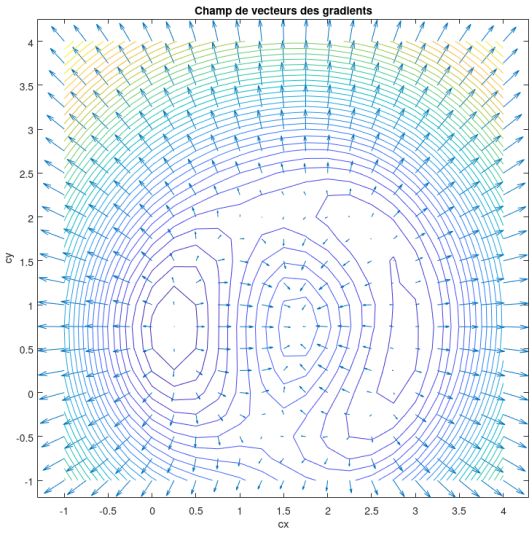
2. Comme à la question 4, on choisit arbitrairement les points $(0;0)$, $(1;1)$, $(2,5;-0,5)$, $(3,5;2)$, $(0;3)$. On calcule le taux d'accroissement avec Matlab dans la fonction `gradient_diff_finie_2(point, delta)`, et on obtient les résultats suivants :

x	y	$\frac{\partial}{\partial x}\mathcal{C}'_{TLS}$	t'_x	$\frac{\partial}{\partial y}\mathcal{C}'_{TLS}$	t'_y	η_x	η_y
0	0	-4.7264	-4.7256	-4.8729	-4.8726	0.0002	0.0001
1	1	6.3604	6.3603	-0.8662	-0.8663	0	0.0001
2.5	-0.5	0.5615	0.5620	-2.8751	-2.8746	0.0009	0.0002
3.5	2	9.9902	9.9904	5.4852	5.4854	0	0
0	3	-6.2627	-6.2626	8.9474	8.9475	0	0

En notant à nouveau $t'_x = \frac{\mathcal{C}'_{TLS}(x+\Delta_x,y) - \mathcal{C}'_{TLS}(x,y)}{\Delta_x}$ et $t'_y = \frac{\mathcal{C}'_{TLS}(x,y+\Delta_y) - \mathcal{C}'_{TLS}(x,y)}{\Delta_y}$.

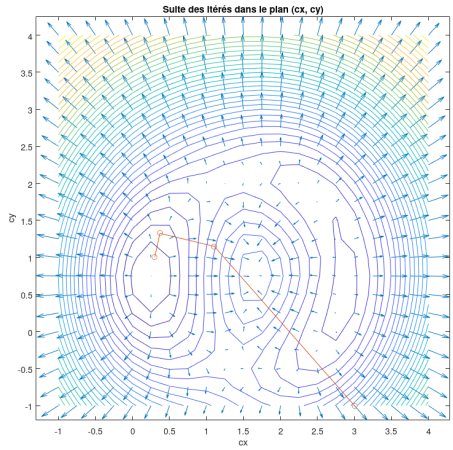
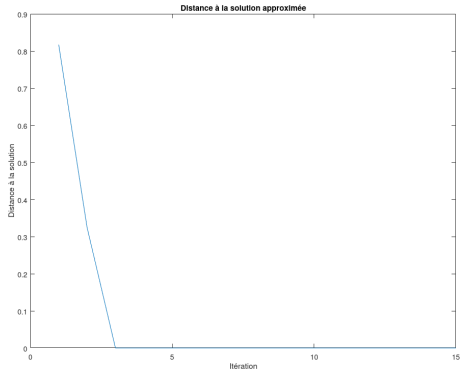
L'erreur relative est, pour les points choisis, systématiquement négligeable.

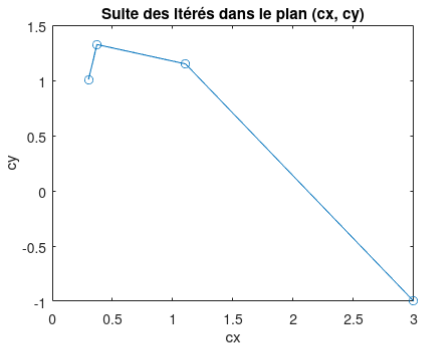
3. En représentant le champ de gradient avec la fonction `quiver`, avec les lignes de contours, on a :

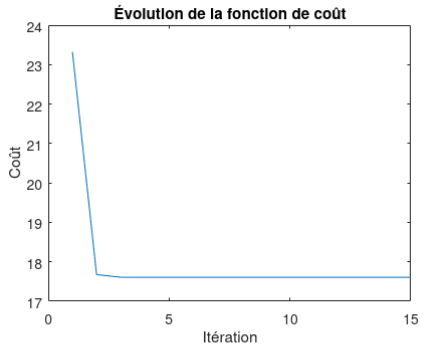


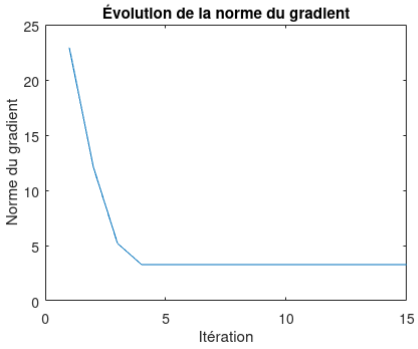
Le gradient est bien orthogonal aux lignes de niveaux. On a prit pour ce gradient une valeur de $\sigma = 1$ en s'appuyant sur les résultats de la question 9 : pour rappel, un σ de 1 garantit d'avoir un bon compromis entre l'efficacité du programme et sa précision.

4. Par la méthode des plus fortes pentes, avec l'algorithme de Fletcher et Lemaréchal, on obtient la solution suivante :

Itérés depuis $(3; -1)$	Distance à la solution
	







5. Comme précédemment, selon le point de départ, on peut tomber dans le mauvais minimum et aboutir à une solution erronée :

Itérés depuis $(0; 0)$	Distance à la solution