OPTIMISATION

Etudes de cas avec utilisation de la toolbox Matlab

Max SONZOGNI
MI5 - Polytech Montpellier

Janvier 2019

1 Algorithmes

Dans la première partie de ce rapport, nous allons étudier différents algorithmes utilisés pour déterminer le minimum local d'une fonction. Ici, les algorithmes qui seront étudiés seront ceux dit du Gradient Simple, Gradient Conjugué et Quasi-Newton.

Le **Gradient Simple** est la méthode qui fournit la pente de descente la plus raide, car elle est de la forme $d = -\nabla f$, ce qui peut donner des cas où l'algorithme va juste tourner autour de la solution sans l'atteindre. Pour pallier ce problème, on ajuste le pas de descente en fonction de l'avancement de la recherche. Pour cela, on utilise un pas optimal t_k^* dont le calcul est détaillé dans l'Etape 3 de l'algorithme suivant :

1. Etape 1 : Initialiser $k = 0$ et $x_k = x_0$
2. Etape 2 : Calculer $d_k = -\nabla f(x_k)$
3. Etape 3 : Calculer le pas optimal $t_k^* = -\frac{\nabla f(x_k)^T d_k}{d_k^T A d_k}$
4. Etape 4 : Calculer $x_{k+1} = x_k + t_k^* d_k$
5. Etape 5 : $k \to k + 1$
6. Etape 6 : Si $k < k_{max}$ et $\|x_{k+1} - x_k\| > \varepsilon_{threshold}$, retour à l'Etape 2 ; Sinon FIN

La méthode du **Gradient Conjugué** est assez similaire à celle du Gradient Simple, la différence étant dans le calcul de la pente d. L'intérêt est ici de corriger la forte raideur de l'algorithme précédent en prenant en compte la pente de descente à l'étape précédente et la matrice hessienne $A = \nabla^2 f$. Cette correction se traduit par l'apparition d'un coefficient β, qui peut se voir dans l'Etape 4 de l'algorithme de calcul :

1. Etape 1 : Initialiser $k = 0$, $x_k = x0$ et $d_k = -\nabla f(x_k)$
2. Etape 2 : Calculer $x_{k+1} = x_k + t_k^* d_k$ avec $t_k^* = -\frac{\nabla f(x_k)^T d_k}{d_k^T A d_k}$
3. Etape 3 : Calculer $\nabla f(x_{k+1})$
4. Etape 4 : Déterminer β_k tel que $\beta_k = \frac{d_k^T A \nabla f(x_{k+1})}{d_k^T A d_k}$
5. Etape 5 : Calculer $d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$
6. Etape 6 : $k \to k + 1$
7. Etape 7 : Si $k < k_{max}$ et $\|d_k\| > \varepsilon_{threshold}$, retour à l'Etape 2 ; Sinon FIN

La méthode **Quasi-Newton** fait partie des méthodes dites de Newton, c'est à dire qu'elle utilise la matrice hessienne de la fonction pour approcher son minimum. Cependant, on utilise ici une approximation de la hessienne au travers de la matrice S. Différentes estimations de cette matrice peuvent être établies (DFP, Correction additive...), mais c'est celle de Broyden-Fletcher-Goldfarb-Shanno (BFGS) qui sera utilisée dans nos tests. L'algorithme de calcul est donc décrit ci-dessous:

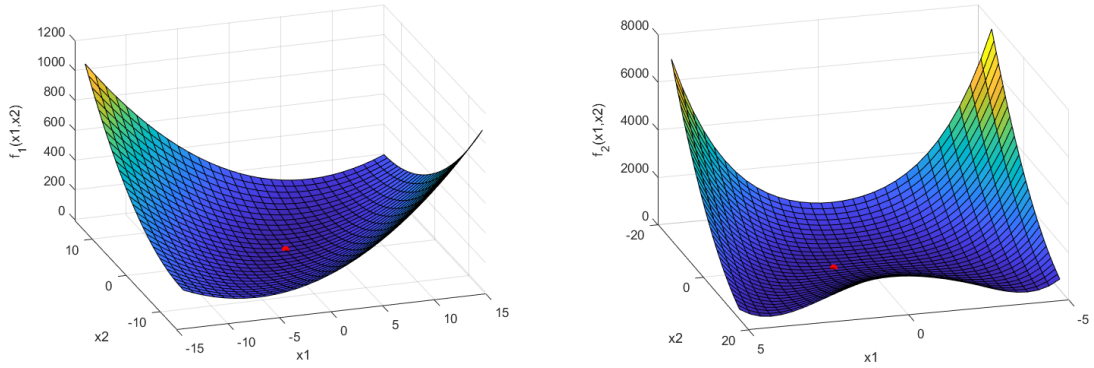
1. Etape 1 : Initialiser $k = 0$, $x_k = x_0$ et $S_k = \mathbb{I}$
2. Etape 2 : Calculer $\nabla f(x_k)$
3. Etape 3 : Calculer $d_k = -S_k \nabla f(x_k)$
4. Etape 4 : Déterminer le pas optimal $t_k^* = -\frac{\nabla f(x_k)^T d_k}{d_k^T A d_k}$
5. Etape 5 : Calculer $x_{k+1} = x_k + t_k^* d_k$ et $\nabla f(x_{k+1})$
6. Etape 6 : Déterminer $\gamma_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ et $\delta_k = x_{k+1} - x_k$

7. Etape 7 : Déterminer S_{k+1} avec la méthode BFGS tel que $S_{k+1} = S_k + \frac{\gamma_k \gamma_k^T}{\gamma_k^T \delta_k} - \frac{S_k \delta_k \delta_k^T S_k}{\delta_k^T S_k \delta_k}$
8. Etape 8 : $k \to k+1$
9. Etape 9 : Si $k < k_{max}$ et $\|x_{k+1} - x_k\| > \varepsilon_{threshold}$, retour à l'Etape 2 ; Sinon FIN

Pour tester ces algorithmes, nous avons choisi 2 fonctions polynomiales, qui sont donc assez simples à dériver, afin pouvoir facilement déterminer leurs gradient et matrice hessienne. Elles sont définies ci-après :

$$f_1(x_1, x_2) = 2x_1^2 + x_2^2 - 2x_1x_2 - x1 - x2 \tag{1}$$

$$f_2(x_1, x_2) = (x_1 - 1)^2 + 5\left(x_1^2 - x_2\right)^2 \tag{2}$$



(a) $f_1(x_1, x_2)$ (b) $f_2(x_1, x_2)$

Figure 1: Représentation 3D des courbes de f_1 et f_2

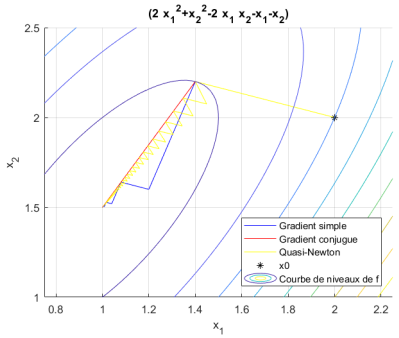
Le point rouge sur les Figures 1a et 1b représente le point donnant le minimum global de chacune des fonctions. Il vaut (1 1.5) pour la première fonction et (1 1) pour la seconde.

Pour pouvoir vérifier si chacune des positions finales x_f renvoyées par les algorithmes sont des minimums locaux ou globaux, les tests seront réalisés en prenant plusieurs position initiales x_0. Les critères d'arrêt k_{max} et $\varepsilon_{threshold}$ seront posés tels que $k_{max} = 10^6$ et $\varepsilon_{threshold} = 10^{-5}$.

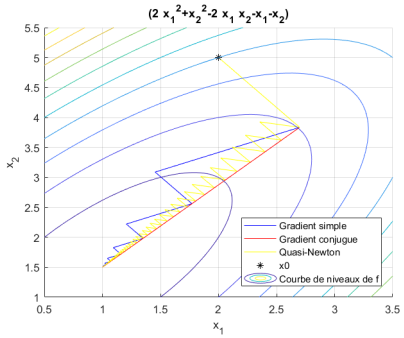
	Gradient Simple			Gradient Conjugué			Quasi-Newton		
x_0	x_f	$f(x_f)$	Itérations	x_f	$f(x_f)$	Itérations	x_f	$f(x_f)$	Itérations
(2 2)	(1 1.5)	-1.25	16	(1 1.5)	-1.25	3	(1 1.5)	-1.25	189
(2 5)	(1 1.5)	-1.25	32	(1 1.5)	-1.25	3	(1 1.5)	-1.25	238
(10 15)	(1 1.5)	-1.25	19	(1 1.5)	-1.25	3	(1 1.5)	-1.25	92
$(-10\ -10)$	(1 1.5)	-1.25	44	(1 1.5)	-1.25	3	(1 1.5)	-1.25	194

Tableau 1: Récapitulatif des valeurs des cas tests avec f_1

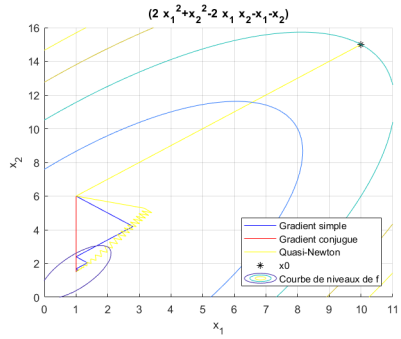
Chaque algorithme converge vers la position minimale $x_f = (1\ 1.5)$ de la fonction f_1, mais tous à un rythme différent. Le Gradient Simple utilise entre 16 et 44 itérations suivant le point x_0 initial pour arriver au point minimum, tandis que le Gradient Conjugué n'en utilise que 3 quelle que soit la position initiale. La méthode de Quasi-Newton prend elle quelques centaines d'itérations pour converger, chose visible sur la Figure 2 car elle fait des zig-zags pour se déplacer, ce qui n'est pas très pratique.



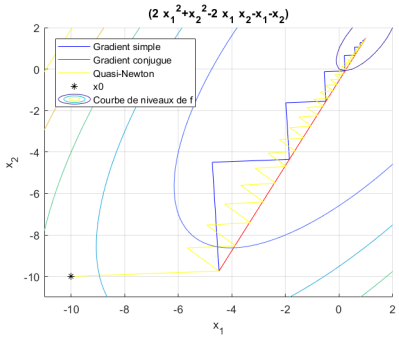
(a) $x_0 = (2\ 2)$



(b) $x_0 = (2\ 5)$

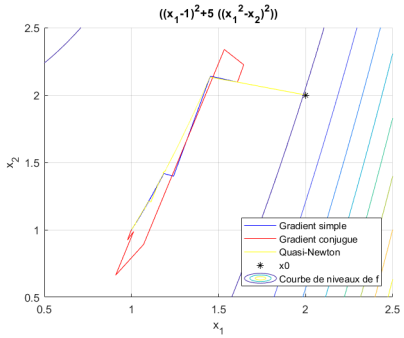


(c) $x_0 = (10\ 15)$

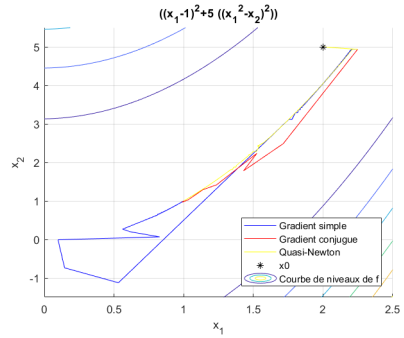


(d) $x_0 = (-10\ \ -10)$

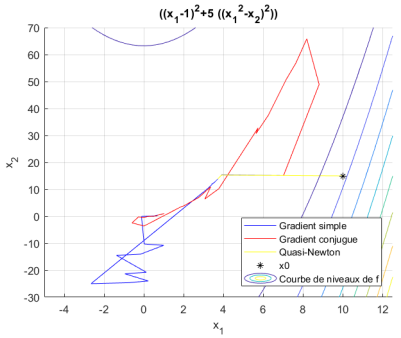
Figure 2: Convergence des différents algorithmes avec $f_1(x_1, x_2)$



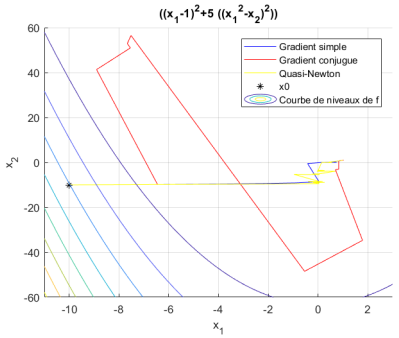
(a) $x_0 = (2\ 2)$



(b) $x_0 = (2\ 5)$



(c) $x_0 = (10\ 15)$



(d) $x_0 = (-10\ \ -10)$

Figure 3: Convergence des différents algorithmes avec $f_2(x_1, x_2)$

	Gradient Simple			Gradient Conjugué			Quasi-Newton		
x_0	x_f	$f(x_f)$	Itérations	x_f	$f(x_f)$	Itérations	x_f	$f(x_f)$	Itérations
(2 2)	(1 1)	0	91	(1 1)	0	13	(1 1)	0	42647
(2 5)	(1 1)	0	448	(1 1)	0	14	(1 1)	0	290216
(10 15)	(1 1)	0	1314	(1 1)	0	34	(3.71 13.8)	7.40	10^6
$(-10\ -10)$	(1 1)	0	310	(1 1)	0	18	(1 1)	0	25438

Tableau 2: Récapitulatif des valeurs des cas tests avec f_2

Ici, c'est toujours la méthode du Gradient Conjugué qui converge le plus rapidement, avec quelques dizaines d'itérations, suivie par la méthode du Gradient Simple, qui lui en nécessite plusieurs centaines. La méthode de Quasi-Newton utilise encore plus d'itérations que précédemment pour converger, et il lui arrive même de ne pas en avoir assez comme dans le cas $x_0 = (10\ 15)$. Cependant elle garde toujours son orientation vers le point minimum contrairement aux deux autres algorithmes.

En conclusion, on peut affirmer que les méthodes de Gradient Simple et Gradient Conjugué utilisent assez peu d'itérations pour converger vers le minimum de la fonction, mais ils ont facilement tendance à diverger quand le gradient est faible. La méthode Quasi-Newton est plutôt lente dans sa convergence, mais sa direction de descente reste toujours proche de la direction vers le point minimum. Ainsi, si l'on cherche à déterminer le chemin le plus court vers le minimum de la fonction, cette méthode peut s'avérer très utile.

2 Dimensionnement de cas réels

Le but de cette section est d'utiliser les outils d'optimisation proposés par Matlab pour obtenir les dimensions idéales dans des cas concrets. Ceux-ci nécessitent de définir 2 nouveaux types de fonctions, les fonctions de coût et fonctions contraintes. Les premières sont similaires à celles utilisées dans le chapitre précédent dans le sens où l'on va en chercher un minimum. La différence se jouera alors sur les fonctions contraintes : comme leur nom l'indique, elles vont imposer des contraintes sur les variables, qui pourront être aussi bien des inégalités que des égalités. On les nommera alors respectivement contraintes d'égalités ou d'inégalités. Ces dernières sont définies de telle sorte que l'algorithme de calcul doive trouver les variables qui respectent $f_{constraint_{ineq}} \leq 0$.
La fonction Matlab permettant d'optimiser ce type de problème est *fmincon*. Tout la documentation nécessaire peut être trouvée dans l'aide Matlab[1].

2.1 Canette

Ici, le choix a été fait de trouver les dimensions idéales de hauteur H et de diamètre D pour une canette (cylindre creux d'épaisseur négligeable) afin qu'elle respecte des contraintes de quantité de matière utilisée. Dans notre cas, on souhaite obtenir une canette qui ait un volume intérieur de 1.5 L tout en minimisant la quantité de matière utilisée, c'est à dire avoir une surface extérieure minimale. Le but est donc d'avoir un volume V supérieur à 1.5 L (fonction contrainte) et une surface S minimale (fonction de coût). Ces fonctions sont donc définies comme telles:

$$f_{constraint_{ineq}} = V - \pi H \left(\frac{D}{2}\right)^2 \tag{3}$$

$$f_{cost} = 2\pi \left(\frac{D}{2}\right)^2 + \pi D H \tag{4}$$

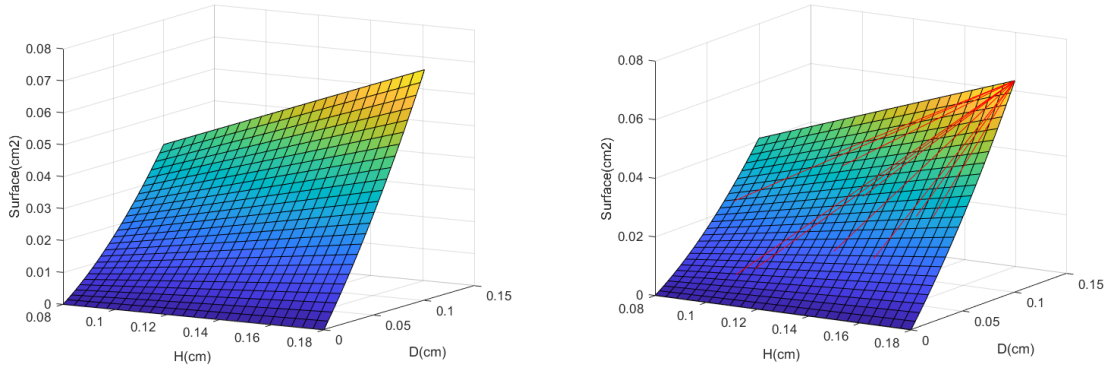
Dans un premier temps, aucune limite n'est imposée sur les dimensions minimales et maximales de hauteur et de rayon. Il se peut donc que la solution optimale rapportée par l'algorithme de calcul donne des dimensions négatives, ce qui serait contraire à la réalité.
Le test a été effectué en prenant 10 couples de valeurs (D, H) au hasard entre 0 et 100 cm. Les valeurs de sortie de l'algorithme sont regroupées dans le tableau 3.

	Valeurs initiales			Valeurs finales		
	D (cm)	H (cm)	Surface (cm^2)	D (cm)	H (cm)	Surface (cm^2)
1	69.90	50.59	18788	0.005	76.30	1.199
2	95.92	89.09	41304	0.005	58.28	1.048
3	13.86	54.72	2684	0.008	29.37	0.744
4	25.75	14.92	2249	0.010	17.13	0.568
5	25.42	84.07	7731	0.124	0.124	0.072
6	24.35	81.42	7161	0.023	3.464	0.260
7	34.99	92.92	12141	-4E8	3E12	-4E21
8	25.10	19.65	2541	0.014	9.469	0.422
9	47.32	61.60	12678	-2E9	3E10	-3E20
10	83.08	35.16	20021	0.124	0.124	0.072

Tableau 3: Valeurs initiales et finales de l'algorithme sans contraintes imposées

On remarque que les valeurs des variables rapportées par l'algorithme d'optimisation varient beaucoup suivant leurs valeurs initiales. La hauteur H des canettes peuvt varier de quelques millimètres à plus d'une dizaine de milliers de kilomètres de haut. De plus, et comme prévu, certains couples donnent des valeurs de diamètre D inférieures à 0, ce qui résulte en une surface négative. Ces résultats ne sont pas physiquement possibles, il est donc nécessaire d'imposer des limitations sur les valeurs que peuvent prendre les variables.

[1] `https://www.mathworks.com/help/optim/ug/fmincon.html`



(a) Surface vierge

(b) Surface + tests

Figure 4: Surface extérieure de la canette en fonction de ses dimensions

On impose maintenant que les dimensions de la canette soient contenues dans un certain intervalle de valeurs. Le choix a été fait de prendre $0 \leq D \leq 10$ cm et $8 \leq H \leq 18$ cm. On peut ainsi tracer une surface 3D représentant la valeur de la fonction de coût en fonction des valeurs de D et de H (c.f. Figure 4a)

Comme précédemment, l'algorithme a été lancé 10 fois en prenant à chaque fois des valeurs initiales au hasard contenues dans les intervalles autorisés. Toutes les valeurs concernant les valeurs initiales et finales sont contenues dans le tableau 4.

	Valeurs initiales				Valeurs finales			
	D (cm)	H (cm)	Surface (cm^2)	Volume (L)	D (cm)	H (cm)	Surface (cm^2)	Volume (L)
1	6.18	13.61	324.5	0.408	9.99	17.99	722.5	1.413
2	2.04	10.34	731.4	0.034	9.99	17.99	722.5	1.413
3	5.10	16.13	299.4	0.329	9.99	17.99	722.5	1.413
4	3.53	13.54	170.1	0.133	9.99	17.99	722.5	1.413
5	2.45	10.90	93.5	0.051	9.99	17.99	722.5	1.413
6	3.20	15.23	169.4	0.122	9.99	17.99	722.5	1.413
7	8.37	11.44	411.3	0.630	9.99	17.99	722.5	1.413
8	5.01	16.81	304.4	0.332	9.99	17.99	722.5	1.413
9	6.59	8.40	242.4	0.287	9.99	17.99	722.5	1.413
10	7.76	12.14	390.8	0.574	9.99	17.99	722.5	1.413

Tableau 4: Valeurs initiales et finales de l'algorithme avec contraintes imposées sur les dimensions

Sur la Figure 4b ont été "*tracées*" les dimensions initiales et finales pour chacun des cas tests.

Il est étonnant de remarquer que chaque couple (D,H) converge vers le couple de valeurs maximales, i.e. (10,18). De plus, on observe que le volume atteint avec ce couple est inférieur à celui désiré, qui était de 1.5 L. Les contraintes appliquées sur les dimensions sont trop restrictives et ne permettent donc pas d'avoir une canette dont le volume est supérieur à 1.5 L. Les solutions permettant de résoudre ce problème seraient soit d'augmenter les valeurs maximales atteignables, soit d'avoir un objectif volumique plus faible.

2.2 Colonne creuse

Le deuxième cas concerne le dimensionnement d'une colonne creuse, de rayon interne R_{int}, de rayon externe R_{ext} et de hauteur L. On souhaite obtenir une masse M minimale pour cette colonne tout en respectant certains critères de résistance, notamment sur la charge maximale et la pression axiale admissibles. Le choix a été fait ici de piloter les rayons de la colonne. On choisit donc de fixer la hauteur L de la colonne à 10 m.

Elle est supposée être réalisée en acier, de masse volumique $\rho = 7850$ kg/m^3 et de module d'Young $E = 210$ GPa. Sa masse, et donc la fonction de coût équivalente, peut donc être déterminée grâce à la formule suivante :

$$M = f_{cost} = \rho\pi L(R_{ext}^2 - R_{int}^2) \tag{5}$$

Contrairement au cas précédent, il y a ici plusieurs fonctions contraintes. La première concerne la charge maximale supportable par la colonne. Celle-ci est définie par la formule :

$$F_C = \frac{\pi^2 EI}{4L^2} \tag{6}$$

avec I le moment quadratique dans l'axe de la colonne. Il faut donc que la charge appliquée F soit inférieure à F_C pour que la contrainte soit respectée.
La deuxième porte sur la pression maximale pouvant être appliquée sur une section de la colonne. La pression appliquée σ est directement reliée à la surface S de la section et à la charge appliquée F par la formule :

$$\sigma = \frac{F}{S} \tag{7}$$

Ainsi, il faut que cette pression appliquée soit inférieure à la pression critique σ_a spécifique au matériau. Une troisième contrainte optionnelle a été ajoutée afin d'avoir des résultats qui soient réalistes. Elle implique que le rayon extérieur de la colonne R_{ext} soit supérieur au rayon intérieur R_{int} plus une petite variation. Cette variation est surtout utile pour ne pas avoir de résultats où les deux rayons sont proches, ce qui rendrait l'épaisseur de la colonne négligeable devant les autres dimensions. Le choix a été fait de prendre cette variation sous la forme d'une proportion de la moyenne des deux rayons, proportion déterminée par une variable ε.

Au final, les fonctions contraintes sont les suivantes :

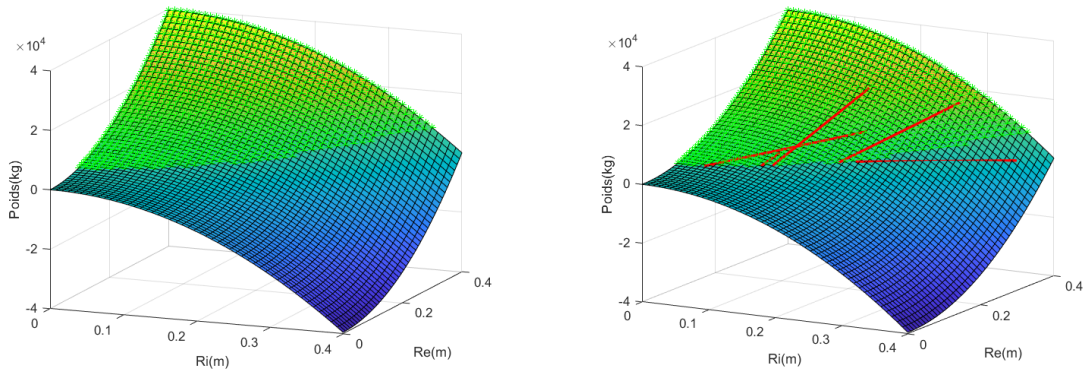
$$f_{constraint_{ineq1}} = F - F_C \tag{8}$$

$$f_{constraint_{ineq2}} = \sigma - \sigma_a \tag{9}$$

$$f_{constraint_{ineq3}} = R_{int} - R_{ext} + \varepsilon\frac{R_{int} + R_{ext}}{2} \tag{10}$$

De plus, on contraint les rayons à être inférieurs à 40 cm pour des raisons d'encombrement. On peut de même que dans le cas précédent tracer la surface 3D de la fonction de coût (c.f. Figure 5).

Dans la première série de tests, on prend les valeurs numériques suivantes de charge $F = 14$ kN, la pression critique maximale $\sigma_a = 0.5$ MPa et la variable ε est posée à 10%. La surface de la fonction de coût représentée dans la Figure 5a comporte des zones où les couples (R_{int},R_{ext}) respectent toutes les fonctions contraintes, désignés par des points verts. Il reste donc à déterminer quels points de ces zones donnent le minimum de la fonction de coût.



(a) Surface + zone admissible

(b) Surface + tests

Figure 5: Poids de la colonne en fonction de ses dimensions

En prenant 5 couples de valeurs (R_{int},R_{ext}) au hasard dans l'intervalle des valeurs possibles, tout en gardant la relation $R_{ext} > R_{int}$, on obtient les valeurs suivantes :

	Valeurs initiales			Valeurs finales			
	R_{int} (cm)	R_{ext} (cm)	M (kg)	R_{int} (cm)	R_{ext} (cm)	M (kg)	S (cm^2)
1	15.98	33.09	20707	11.49	14.87	2198.01	279.9
2	36.38	36.40	24.98	19.99	22.10	2197.98	277.7
3	27.16	37.57	16625	18.28	20.58	2198.01	280.7
4	10.97	17.73	4790	10.25	13.93	2198.00	279.5
5	18.82	27.28	9621	3.80	10.17	2198.00	279.5

Tableau 5: Valeurs initiales et finales de l'algorithme pour la colonne creuse

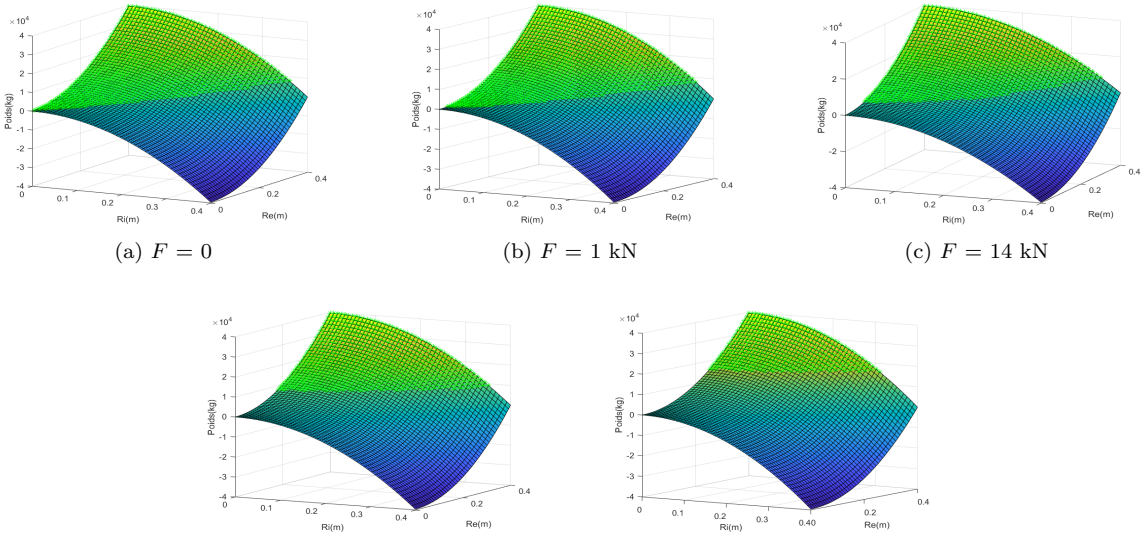
Le tableau 5 permet de voir que la masse finale atteinte est quasiment la même, environ 2200 kg, quel que soit le couple (R_{int},R_{ext}) choisi initialement. La différence réside dans les couples de rayons finaux : le rayon interne R_{int} peut varier entre 3 et 20 cm tandis que le rayon externe R_{ext} varie lui entre 10 et 22 cm. Ainsi, les colonnes peuvent être plus ou moins creuses, mais la surface de leur section tourne autour de 280 cm^2. Cela peut être utile si l'on souhaite faire passer des câbles ou d'autres choses à l'intérieur de la colonne, le seul critère à respecter étant qu'il faille que la surface de la section soit proche de 280 cm^2.

2.2.1 Variations de la charge appliquée F

Dans un second temps, on souhaite observer l'impact qu'ont la charge appliquée F et la pression critique σ_a sur les solutions de ce problème.

Afin de déterminer quel est l'impact dans un permier temps de la charge appliquée F, plusieurs tests sont effectués en faisant varier la valeur de cette charge tout en gardant constantes les autres variables. Ces tests permettent alors de tracer les zones admissibles de solutions, qui sont toutes regroupées dans la Figure 6.

Ces surfaces montrent une tendance assez claire : plus la charge appliquée augmente, plus les valeurs admissibles des couples (R_{int},R_{ext}) sont grandes. Ces observations peuvent être démontrés grâce à l'équation 7. En effet, plus la charge appliquée est élevée, plus la surface requise pour avoir la même pression doit être grande, et donc soit les rayons de la colonne doivent être grands soit il doit y avoir une large différence entre le rayon intérieur et le rayon extérieur.



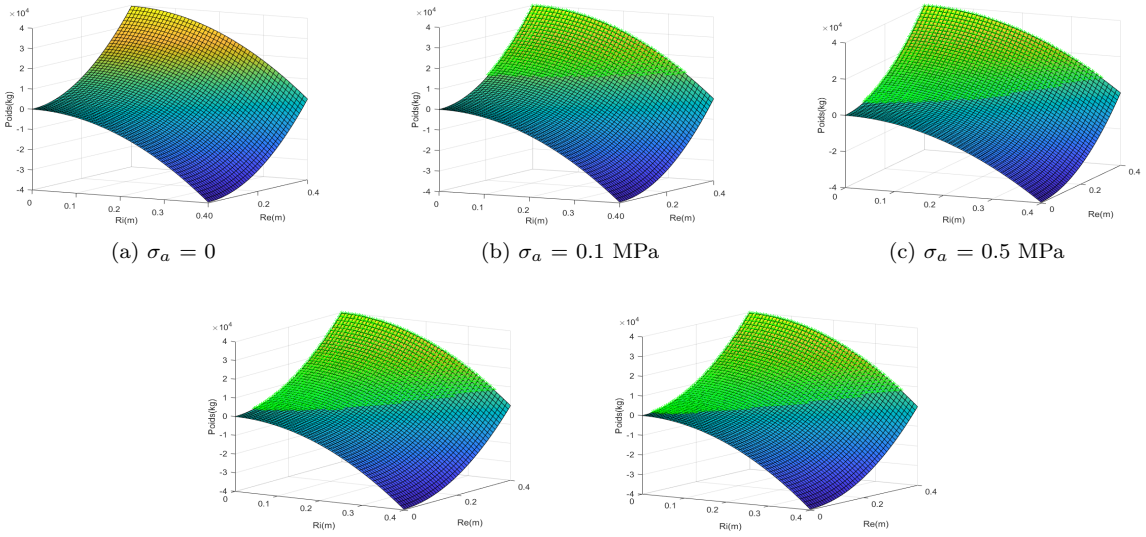
(a) $F = 0$ (b) $F = 1$ kN (c) $F = 14$ kN

(d) $F = 50$ kN (e) $F = 100$ kN

Figure 6: Zone admissible en fonction de la charge appliquée F

2.2.2 Variations de la pression critique σ_a

Les mêmes tests sont ensuite réalisés sur la pression critique σ_a, dont les zones admissibles sont ensuite présentées dans la Figure 7.



(a) $\sigma_a = 0$ (b) $\sigma_a = 0.1$ MPa (c) $\sigma_a = 0.5$ MPa

(d) $\sigma_a = 1$ MPa (e) $\sigma_a = 10$ MPa

Figure 7: Zone admissible en fonction de la pression critique σ_a

Ici, les zones admissibles de solutions augmentent plus la pression critique est élevée. Cela peut aussi être déduit de l'équation 7. En effet, à charge appliquée égale, plus la pression maximale est élevée, plus la surface est étroite, et donc les rayons internes et externes de la colonne peuvent être petits. De plus, quand σ_a est nul, il n'y a aucun couple de valeurs (R_{int},R_{ext}) qui peut être solution. Quand la pression maximale est nulle, cela veut dire que la colonne ne peut supporter aucune pression dans son axe, et donc il faut une surface "*infinie*" pour y parvenir, ce qui est impossible.

3 Optimisation de forme

Cette section va se concentrer sur l'optimisation de la forme d'une corde dans un plan 2D. Cette corde peut être modélisée par une courbe de longueur L et de rigidité de flexion R_f. Dans la suite de l'étude, ces paramètres seront fixes et vaudront $L = 1$ m et $R_f = 1$ N.m^2. La distance d'un point le long de cette courbe est décrite par son abscisse curviligne s telle que $0 \leq s \leq L$. On peut alors décrire la position de ce point dans l'espace 2D grâce à l'équation :

$$p(s) = \int_0^s \begin{pmatrix} cos\ \theta(s) \\ sin\ \theta(s) \end{pmatrix} ds \tag{11}$$

où $\theta(s)$ représente l'angle d'orientation de la courbe. L'angle d'orientation est lui décomposé sous la forme d'une série de Fourier, qui s'écrit alors :

$$\theta(s) = a_1 + a_2 s + a_3 sin\left(\frac{2\pi}{L}s\right) + a_4 cos\left(\frac{2\pi}{L}s\right) + a_5 sin\left(\frac{4\pi}{L}s\right) + a_6 cos\left(\frac{4\pi}{L}s\right) \tag{12}$$

avec $a_i \in \mathbb{R}^6$, i =1,...,6 les coefficients de cette série. On impose que la courbe démarre au point $p_0 = p(s = 0) = O$ avec une orientation $\theta_0 = \theta(s = 0)$ et se termine au point $p_1 = p(s = L)$ avec un angle d'orientation $\theta_1 = \theta(s = L)$. Ce seront donc ces égalités qui formeront les fonctions contraintes du problème d'optimisation.

Dans ce problème, on cherche à ce que l'énergie potentielle de la courbe soit minimale, c'est à dire qu'elle fasse le moins de virages possibles. L'énergie potentielle U est donnée par l'équation suivante :

$$U(a) = \frac{1}{2}\int_0^L R_f \left(\frac{d\theta}{ds}\right)^2 ds \tag{13}$$

Dans un premier temps, on choisit comme contraintes de positionnement $\theta_0 = 0$, $\theta_1 = \frac{\pi}{4}$ et $p_1 = (0.5\ 0.5)^T$. En faisant 100 essais à partir de valeurs initiales de a au hasard entre 0 et 1, on peut exprimer les résultats sous forme de probabilité d'apparition de chaque forme de courbe. Ainsi, on obtient les résultats suivants :

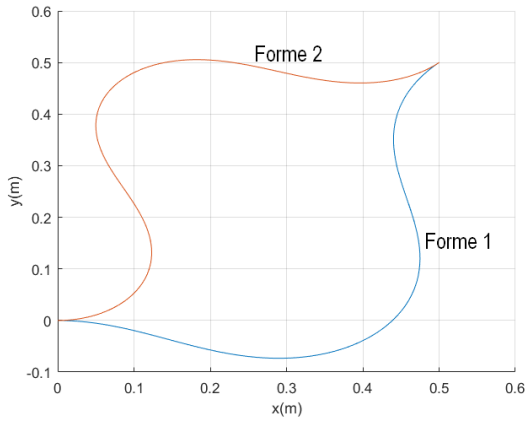


Figure 8: Formes de la courbe données par l'algorithme d'optimisation

	a_1	a_2	a_3	a_4	a_5	a_6	U (N.m)	Nb apparitions
1	0.373	0.785	-0.796	-0.351	-0.027	-0.022	7.828	21
2	0.407	0.785	1.259	-0.444	-0.053	0.036	18.08	79

Tableau 6: Récapitulatif des valeurs renvoyées par l'algorithme

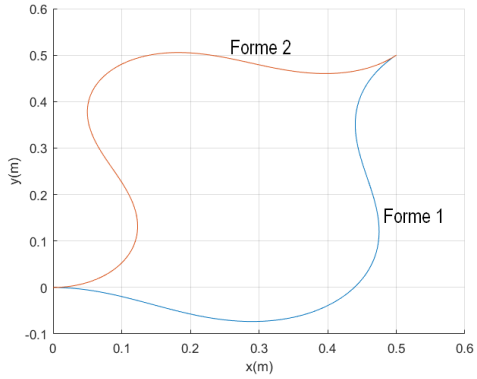
Avec les contraintes sélectionnées pour ce cas-ci, il n'y a que 2 configurations de la courbe qui sont possibles (c.f. Figure 8). La première a une fréquence d'apparition inférieure à la seconde, mais c'est

aussi elle qui possède l'énergie potentielle la plus basse. La fréquence d'apparition dépend beaucoup des conditions initiales choisies, ce qui peut donner, dans des situations où il n'y a que très peu de cas tests, une courbe qui n'est pas celle avec l'énergie potentielle la plus faible, et donc une solution à ce problème qui n'est pas optimale.

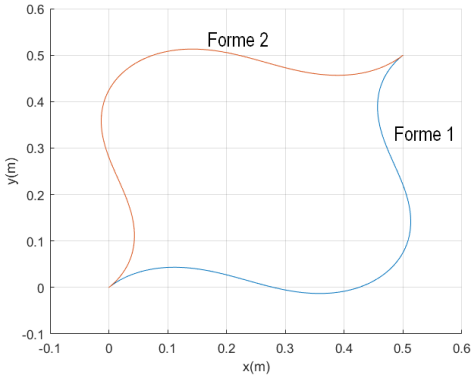
De même que dans l'étude précédente, nous allons étudier l'effet qu'ont les contraintes θ_0, θ_1 et p_1 sur les formes de la corde possibles.

3.1 Variations de θ_0

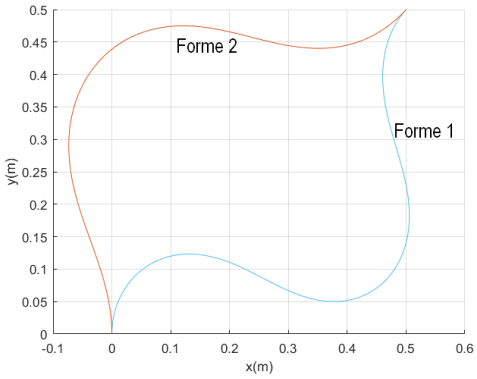
Nous allons en premier lieu observer quels sont les effets de l'angle d'orientation initial θ_0 sur les courbes. On choisit de le faire varier de 0 à $\frac{3\pi}{4}$ par pas de $\frac{\pi}{4}$. Les résultats obtenus sont regroupés dans le Tableau 7 et les courbes y correspondant sont visibles dans la Figure 9.



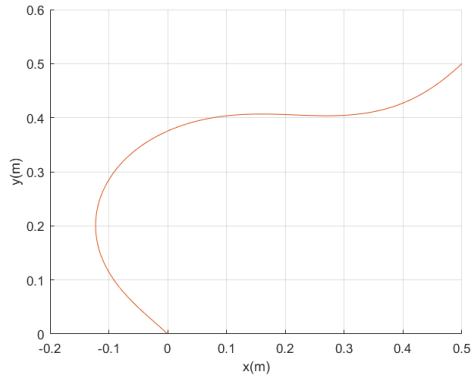
(a) $\theta_0 = 0$



(b) $\theta_0 = \frac{\pi}{4}$



(c) $\theta_0 = \frac{\pi}{2}$



(d) $\theta_0 = \frac{3\pi}{4}$

Figure 9: Formes possibles de la courbe en fonction de θ_0

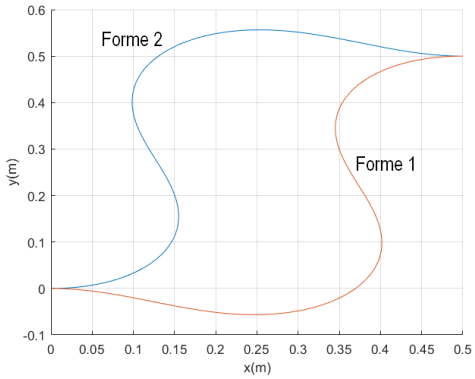
Quand θ_0 est trop excentré par rapport à la direction $\overrightarrow{p_0 p_1}$ (Figure 9d), il n'y a qu'une seule forme de la courbe minimisant l'énergie potentielle qui est possible. Si θ_0 et θ_1 sont exactement dans l'axe (Figure 9b), les 2 formes de la corde sont symétriques par rapport à cet axe. Leur seule différence se fait sur le coefficient a_3, qui est opposé entre la courbe 1 et la courbe 2. Cependant, il y a toujours cet écart dans le nombre d'apparitions, même dans le cas de courbes symétriques. Il est donc évident que choisir les valeurs initiales des a_i entre 0 et 1 résulte dans ces différences.

θ_0		a_1	a_2	a_3	a_4	a_5	a_6	U (N.m)	Nb apparitions
0	1	0.373	0.785	-0.796	-0.351	-0.027	-0.022	7.828	21
	2	0.407	0.785	1.259	-0.444	-0.053	0.036	18.08	79
$\frac{\pi}{4}$	1	0.785	0	1.126	0	0	0	12.52	85
	2	0.785	0	-1,126	0	0	0	12.52	15
$\frac{\pi}{2}$	1	1.163	-0.785	-1.259	0.444	0.053	-0.036	18.08	15
	2	1.197	-0.785	0.796	0.351	0.027	0.022	7.828	85
$\frac{3\pi}{4}$	1	1.646	-1.570	0.363	0.618	0.039	0.091	6.705	100

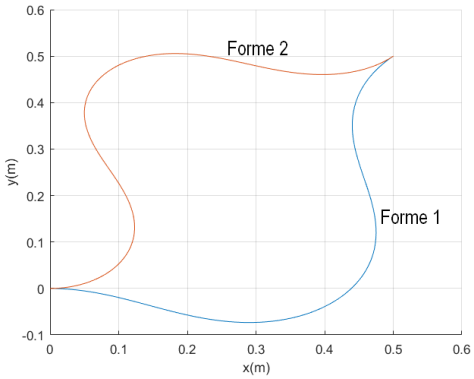
Tableau 7: Récapitulatif des coefficients pour différentes valeurs de θ_0

3.2 Variations de θ_1

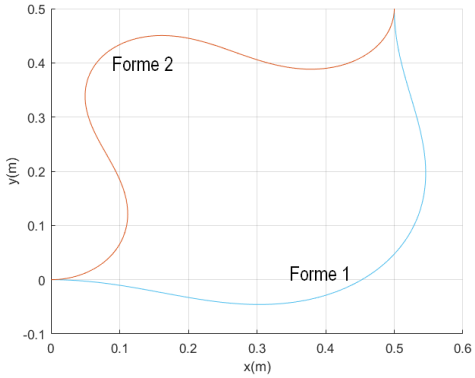
Après avoir étudié les effets de θ_0 sur les courbes possibles vient le tour de θ_1, l'angle d'inclinaison autour du point p_1. Ce sont les mêmes test que pour θ_0 qui ont été effectués pour pouvoir mieux les comparer. Les résultats peuvent être trouvés dans le Tableau 8 et dans la Figure 10.



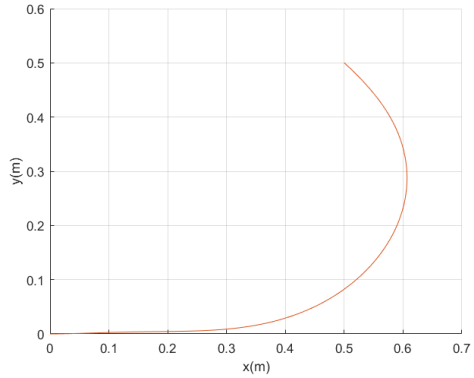
(a) $\theta_1 = 0$



(b) $\theta_1 = \frac{\pi}{4}$



(c) $\theta_1 = \frac{\pi}{2}$



(d) $\theta_1 = \frac{3\pi}{4}$

Figure 10: Formes possibles de la courbe en fonction de θ_1

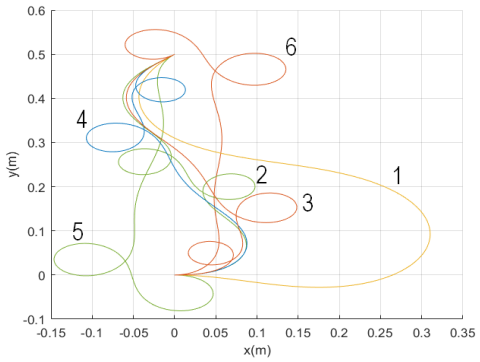
Les mêmes commentaires que ceux établis pour θ_0 peuvent être faits pour l'influence de θ_1 : si les courbes ne sont pas symétriques, il y en a toujours une qui possède une énergie potentielle interne inférieure à l'autre, et si l'angle d'inclinaison est trop décentré, une seule forme de courbe est possible.

θ_1		a_1	a_2	a_3	a_4	a_5	a_6	U (N.m)	Nb apparitions
0	1	0.785	0	-0.807	-0.785	0	0	12.521	22
	2	0.785	0	0.807	-0.785	0	0	12.521	78
$\frac{\pi}{4}$	1	0.373	0.785	-0.796	-0.351	-0.027	-0.022	7.828	21
	2	0.407	0.785	1.259	-0.444	-0.053	0.036	18.08	79
$\frac{\pi}{2}$	1	0	1.570	-0.538	0	-0.031	0	4.130	26
	2	0	1.570	1.514	0	-0.138	0	24.63	74
$\frac{3\pi}{4}$	1	-0.334	2.356	-0.165	0.282	-0.019	0.051	3.953	100

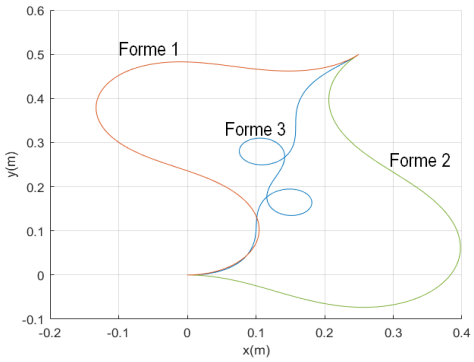
Tableau 8: Récapitulatif des coefficients pour différentes valeurs de θ_1

3.3 Variations de p_1

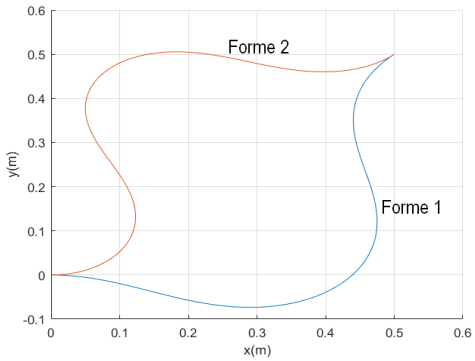
Pour terminer ce sera l'influence du point p_1, et donc de la distance entre les points de départ et d'arrivée, qui va être étudiée. On choisit cette fois de déplacer le point p_1 le long de l'axe x entre 0 et 0.75 par pas de 0.25 m.



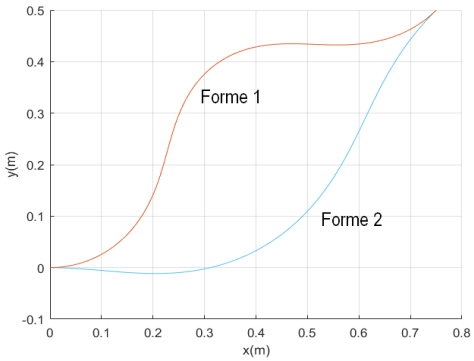
(a) $p_1 = (0\ 0.5)^T$



(b) $p_1 = (0.25\ 0.5)^T$



(c) $p_1 = (0.5\ 0.5)^T$



(d) $p_1 = (0.75\ 0.5)^T$

Figure 11: Formes possibles de la courbe en fonction de p_1

Il est important de remarquer que lorsque la distance entre les points de départ et d'arrivée est faible, il y a plus de formes de courbes possibles (Figure 11a et 11b). Il y a une forme majoritaire, celle avec le minimum d'énergie potentielle, qui est plutôt banale tandis que les autres sont plus "*fantaisistes*" et forment des boucles. Ces boucles sont la raison pourquoi leur énergie est bien plus élevée, car les virages l'augmentent considérablement. La forme 2 de la Figure 11d en est un très bon exemple: elle n'est composée que d'un seul "*tournant*" et c'est elle qui possède l'énergie la plus faible de tous les cas tests effectués (voir Tableau 9).

p_1		a_1	a_2	a_3	a_4	a_5	a_6	U (N.m)	Nb apparitions
$\begin{pmatrix} 0 \\ 0.5 \end{pmatrix}$	1	1.086	0.785	-0.874	-1.022	-0.041	-0.063	18.39	89
	2	-0.486	0.785	0.120	2.400	0.191	-1.914	203.4	6
	3	3.047	0.785	2.605	-1.889	-1.087	-1.158	202.1	2
	4	3.068	0.785	-2.168	-1.986	1.312	-1.081	199.9	1
	5	-0.896	0.785	-2.718	-0.206	-0.108	1.103	122.1	1
	6	-0.716	0.785	3.087	-0.376	0.343	1.093	147.5	1
$\begin{pmatrix} 0.25 \\ 0.5 \end{pmatrix}$	1	0.757	0.785	1.381	-0.816	-0.053	0.058	25.97	78
	2	0.664	0.785	-0.998	-0.631	-0.030	-0.032	14.17	21
	3	-1.079	0.785	-0.054	2.738	0.383	-1.658	188.8	1
$\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$	1	0.373	0.785	-0.796	-0.351	-0.027	-0.022	7.828	21
	2	0.407	0.785	1.259	-0.444	-0.053	0.036	18.08	79
$\begin{pmatrix} 0.75 \\ 0.5 \end{pmatrix}$	1	0.198	0.785	0.780	-0.242	-0.071	0.044	7.181	70
	2	0.190	0.785	-0.323	-0.169	-0.020	-0.020	1.661	30

Tableau 9: Récapitulatif des coefficients pour différentes valeurs de p_1

Le fait de choisir les conditions initiales entre 0 et 1 en est pour beaucoup dans la probabilité d'apparition de chacune des solutions. Ceci est facilement démontrable : en prenant $\theta_0 = \theta_1 = 0$ et $p_1 = p_0 = (0\ 0)^T$, on obtient la Figure 12. On peut y voir que la même forme est répétée pour chacun des cas tests, et qu'il devrait y en avoir autant en haut qu'en bas si les conditions initiales permettait une répartition équilibrée des solutions. Or il y a plus de courbes dans la partie haute de la figure, ce qui veut dire que le choix des conditions initiales n'est pas optimal. Il pourrait donc être judicieux de trouver l'intervalle donnant une équiprobabilité des solutions. Ce travail pourra être réalisé dans une autre étude.

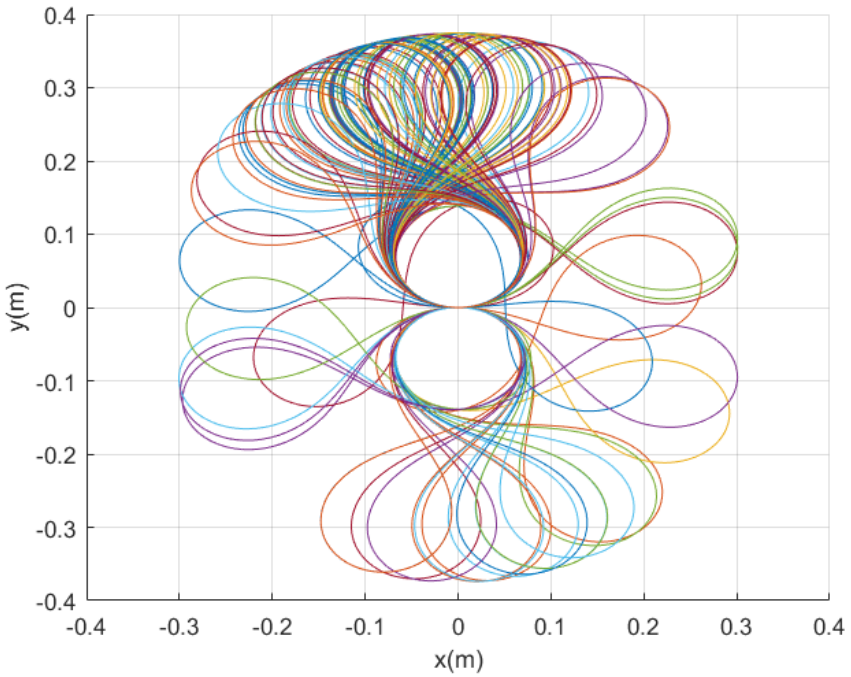


Figure 12: Répartition des courbes pour $\theta_0 = \theta_1 = 0$ et $p_1 = p_0 = (0\ 0)^T$

4 Robotique & Optimisation multi-objectifs

Dans cette partie, nous allons étudier un robot symétrique à 2 bras. Le but sera de positionner ses bras à un endroit désiré tout en imposant aussi une position sur son centre de masse.

La position du bras gauche du robot est donnée par :

$$\begin{aligned} x_L &= -l_1 s_1 + l_e c_1 + l_2 c_{12} + l_3 c_{123} \\ y_L &= l_1 c_1 + l_e s_1 + l_2 s_{12} + l_3 s_{123} \end{aligned} \tag{14}$$

tandis que la position du bras droit est :

$$\begin{aligned} x_R &= -l_1 s_1 - l_e c_1 - l_4 c_{14} - l_5 c_{145} \\ y_R &= l_1 c_1 - l_e s_1 - l_4 s_{14} - l_5 s_{145} \end{aligned} \tag{15}$$

avec $c_i = cos(\theta_i)$, $s_i = sin(\theta_i)$, $c_{ij} = cos(\theta_i + \theta_j)$ et $s_{ij} = sin(\theta_i + \theta_j)$.

La position du centre de masse du robot est donnée par :

$$\begin{aligned} x_{CdM} &= -\frac{1}{M}(r_e c_1 + r_1 s_1 + r_2 c_{12} + r_3 c_{123} + r_4 c_{14} + r_5 c_{145}) \\ y_{CdM} &= \frac{1}{M}(-r_e s_1 + r_1 c_1 - r_2 s_{12} - r_3 s_{123} - r_4 s_{14} - r_5 s_{145}) \end{aligned} \tag{16}$$

avec

$$\begin{aligned} r_e &= l_e(m_4 + m_5 - m_3 - m_2) \\ r_1 &= l_1(m_2 + m_3 + m_4 + m_5) + l_{1m} m_1 \\ r_2 &= -l_2 m_3 - l_{2m} m_2 \\ r_3 &= -l_{3m} m_3 \\ r_4 &= l_4 m_5 + l_{4m} m_4 \\ r_5 &= l_{5m} m_5 \\ M &= m_1 + m_2 + m_3 + m_4 + m_5 \end{aligned} \tag{17}$$

et m_i la masse du segment i et l_{im} la position du centre de masse du segment i.

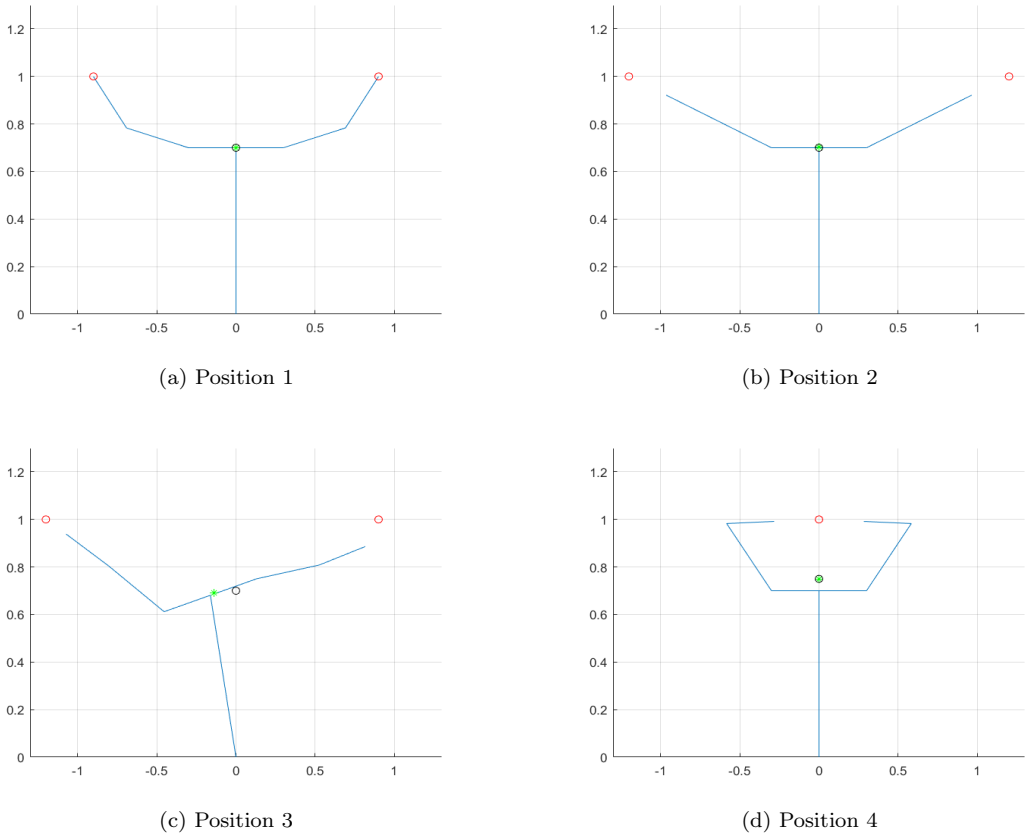
Les fonctions de coût sont ici égales à la norme de la distance entre la position du robot et la position désirée. Elle sont donc définies comme $\varepsilon_i = \left\| x_i - x_i^d \right\|$. On impose $-\frac{\pi}{6} \leq \theta_1 \leq \frac{\pi}{6}$, $-\frac{\pi}{4} \leq \theta_{2,4} \leq \frac{\pi}{4}$ et $-\frac{5\pi}{6} \leq \theta_{3,5} \leq \frac{5\pi}{6}$, et comme dimensions du robot :

- $l_1 = 0.7$ m, $l_e = 0.3$ m, $l_2 = l_4 = 0.4$ m, $l_3 = l_5 = 0.3$ m
- $m_1 = 20$ kg, $m_2 = m_4 = 6$ kg, $m_3 = m_5 = 4$ kg
- $l_{1m} = 0.6$ m, $l_{2m} = l_{4m} = 0.2$ m, $l_{3m} = l_{5m} = 0.15$ m

Pour effectuer tous les tests suivants, on prendra comme condition initiale $\theta_0 = [0\ 0\ 0\ 0\ 0]$ sur les angles du robot. Ces valeurs donnent le robot avec une forme de T (base verticale et bras allongés horizontalement). Une série de 4 tests avec différentes positions à atteindre pour les bras et le centre de masse ont alors été réalisés, dont les résultats sont visibles ci-dessous :

	x_R^d	x_R	ε_R	x_L^d	x_L	ε_L	x_{CdM}^d	x_{CdM}	ε_{CdM}
1	$\begin{pmatrix} -0.9 \\ 1 \end{pmatrix}$	$\begin{pmatrix} -0.89 \\ 0.99 \end{pmatrix}$	10^{-4}	$\begin{pmatrix} 0.9 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0.89 \\ 0.99 \end{pmatrix}$	10^{-4}	$\begin{pmatrix} 0 \\ 0.7 \end{pmatrix}$	$\begin{pmatrix} 10^{-7} \\ 0.70 \end{pmatrix}$	10^{-4}
2	$\begin{pmatrix} -1.2 \\ 1 \end{pmatrix}$	$\begin{pmatrix} -0.96 \\ 0.92 \end{pmatrix}$	0.248	$\begin{pmatrix} 1.2 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0.96 \\ 0.92 \end{pmatrix}$	0.248	$\begin{pmatrix} 0 \\ 0.7 \end{pmatrix}$	$\begin{pmatrix} 10^{-13} \\ 0.70 \end{pmatrix}$	10^{-3}
3	$\begin{pmatrix} -1.2 \\ 1 \end{pmatrix}$	$\begin{pmatrix} -1.07 \\ 0.94 \end{pmatrix}$	0.141	$\begin{pmatrix} 0.9 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0.82 \\ 0.89 \end{pmatrix}$	0.141	$\begin{pmatrix} 0 \\ 0.7 \end{pmatrix}$	$\begin{pmatrix} -0.14 \\ 0.69 \end{pmatrix}$	0.141
4	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} -0.28 \\ 0.99 \end{pmatrix}$	0.283	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0.28 \\ 0.99 \end{pmatrix}$	0.283	$\begin{pmatrix} 0 \\ 0.75 \end{pmatrix}$	$\begin{pmatrix} 10^{-12} \\ 0.74 \end{pmatrix}$	10^{-4}

Tableau 10: Valeurs des positions du robot correspondant à la Figure 13



(a) Position 1

(b) Position 2

(c) Position 3

(d) Position 4

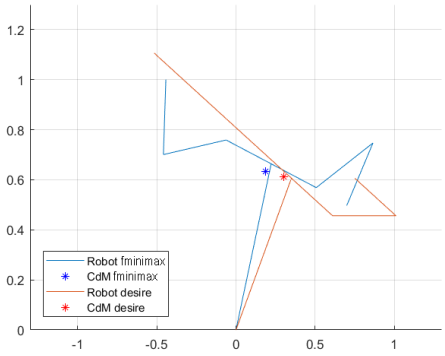
Figure 13: Positions du robot suivant les valeurs imposées dans le Tableau 10

Dans le cas 1 (Figure 13a), les positions imposées au robot (cercles rouges pour les bras et cercle noir pour le centre de masse) sont toutes atteignables en même temps. La fonction d'optimisation y arrive facilement et donne une erreur sur chacune des positions de l'ordre du dixième de millimètre, ce qui correspond à un placement quasiment parfait.

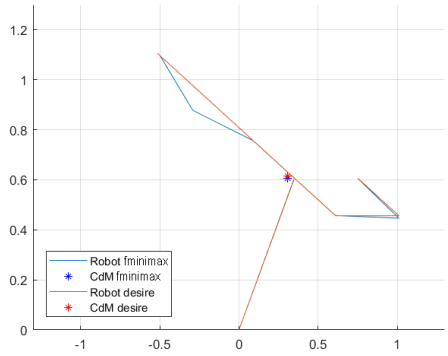
Pour le deuxième cas, les positions désirées pour les bras ne sont pas atteignables en même temps, tandis que le centre de masse (représenté par une étoile verte) est fixé au centre : le robot est alors symétrique. Cette symétrie donne une erreur sur le centre de masse relativement faible comparée à celle sur les bras. L'algorithme préfère donc avoir une seule fonction de coût quasiment minimale et les autres assez élevée plutôt que toutes avec la même valeur non minimale.

Le troisième test impose une disymétrie sur les bras tout en gardant le centre de masse au centre. Le robot est alors obligé de pencher vers le côté où la position désirée du bras est la plus éloignée (ici côté droit), et donc le centre de masse se place de ce côté-ci. Pour ce cas-là, les erreurs sur les positions sont toutes égales, ce qui montre que la fonction d'optimisation fait en sorte d'avoir le même minimum sur chacune des fonctions de coût quand il ne peut pas en favoriser une par rapport aux autres (voir cas 2 Figure 13b).

Sur la Figure 13d, le robot essaie d'atteindre avec ses deux bras la position centrale, chose qui lui est impossible. En effet, cette position symétrique n'est possible qu'en dépassant les angles limites qui ont imposés. Il doit alors se placer de façon à être le plus proche possible de ses objectifs, ce qui fait que les angles de ses bras correspondent aux valeurs limites ($\theta_2 = -\theta_4 = \frac{\pi}{4}$, $\theta_3 = -\theta_5 = \frac{5\pi}{6}$).



(a) $\theta_0 = [0\ 0\ 0\ 0\ 0]$



(b) $\theta_0 = [-\frac{\pi}{6}\ 0\ 0\ 0\ 0]$

Figure 14: Positions du robot en fonction des conditions initiales

Un dernier test a été effectué pour vérifier si l'algorithme d'optimisation permettait d'atteindre une position possible quelle qu'elle soit. Le choix a été fait de prendre $\theta^d = [-\frac{\pi}{6}\ \frac{\pi}{6}\ \frac{5\pi}{6}\ 0\ 0]$ afin d'avoir approximativement comme positions à atteindre $x_R^d = (-0.5\ 1.1)^T$, $x_L^d = (0.75\ 0.6)^T$ et $x_{CdM}^d = (-0.3\ 0.6)^T$. Après avoir lancé la fonction d'optimisation *fminimax*[2], les valeurs de θ retournées permettent de tracer le robot, visualisable sur la Figure 14a.

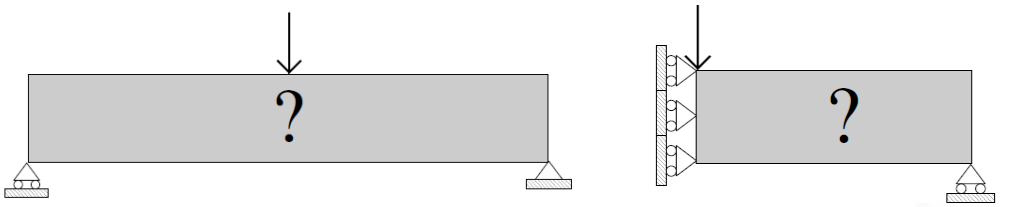
Il peut alors être observé que quand bien même les positions des bras et du CdM sont atteignables en même temps, aucune d'entre elles n'est obtenue : la différence de positionnement est égale à 0.115 m pour chacune des fonctions de coût. Le programme "*préfère*" avoir des valeurs des fonctions de coût toujours décroissantes pendant son exécution plutôt que devoir subir une augmentation de ces fonctions pour pouvoir atteindre le minimum global. On peut donc en conclure que cet algorithme permet d'atteindre seulement un minimum local s'il utilise plusieurs fonctions de coût, même s'il existe un minimum global pour ce même problème.

Ainsi, en prenant comme conditions initiales $\theta_0 = [-\frac{\pi}{6}\ 0\ 0\ 0\ 0]$, le robot atteint presque la position désirée (c.f. Figure 14b). Dans ce cas-ci, l'erreur de positionnement ε n'est que de 4.10^{-3} sur chaque placement, mais on peut voir que le bras droit du robot n'est pas exactement positionné. Cela est dû à la mauvaise définition des positions à atteindre : celles-ci ne sont que des approximations des positions obtenues avec les angles θ^d définis au début du test, et ne correspondent pas exactement à celles du robot désiré sur la Figure 14.

[2] `https://www.mathworks.com/help/optim/ug/fminimax.html`

5 Optimisation topologique

Un article de Ole Sigmund paru en 2000[3] propose un code Matlab permettant de trouver la forme optimale d'une poutre fixée en ses extrémités et où une force est appliquée au centre de sa partie supérieure (voir Figure 15). Ce programme se base sur une analyse Elements Finis et un algorithme d'optimisation pour arriver à déterminer la forme que doit prendre cette poutre pour que ses déplacements totaux sont minimaux.



(a) Poutre complète (b) Demi-poutre (symétries des CL)

Figure 15: Illustration de la poutre et des conditions limites appliquées

Le problème d'optimisation doit minimiser la souplesse c (qui est l'inverse de la rigidité) de la structure tout en respectant 3 contraintes : il faut que la proportion volumique de matière dans la poutre soit toujours égale à `volfrac`; chaque élément doit avoir une proportion volumique comprise entre une valeur minimale `xmin` (ici posée à 10^{-3}) et 1; le champ de déplacements U des éléments de la poutre doit être solution du problème Elements Finis $KU = F$.

La poutre est d'abord décomposée en `nelx` éléments sur sa longueur et en `nely` sur sa largeur. On impose alors que chaque élément soit rempli avec une proportion de matière `volfrac` spécifiée manuellement. Une fois le calcul Elements Finis réalisé sur cette première structure et les déplacements sur chaque élément déterminés, l'algorithme d'optimisation choisit s'il faut rajouter plus ou moins de matière sur chacun d'entre eux. Pour cela, il recalcule dans un premier temps $\frac{\partial c}{\partial x_e}$ afin de mieux trouver les zones où la quantité de matière doit être modifiée.

Ensuite vient la partie optimisation à proprement parler : les coefficients x_e pour chaque élément e de la structure sont réévalués suivant le déplacement qu'ils subissent. La nouvelle valeur x_e^{new} est déterminée parmi 3 cas suivant la valeur de x_e: $max(x_{min}, x_e - m) < x_e B_e^{\eta} < min(1, x_e + m)$ avec m une valeur limite positive, $\eta = \frac{1}{2}$, $B_e = \frac{1}{\lambda} \frac{-\partial c}{\partial x_e}$ et λ un multiplicateur de Lagrange recalculé tout au long de l'exécution de l'algorithme d'optimisation par une méthode de dichotomie.

Le programme relance alors un calcul Elements Finis sur la nouvelle structure, sort un nouveau champ de déplacements, remodifie les coefficients x_e, tout ceci en boucle jusqu'à ce qu'il trouve une forme stabilisée. Le programme s'arrête alors lorsque la différence maximale entre x_e^i et x_e^{i+1} est inférieure à 1%.

Des tests ont été lancés pour voir quelle était l'influence dela proportion volumique `volfrac`, du coefficient de pénalisation `penal` et du rayon minimum `rmin` sur les formes données par le programme.

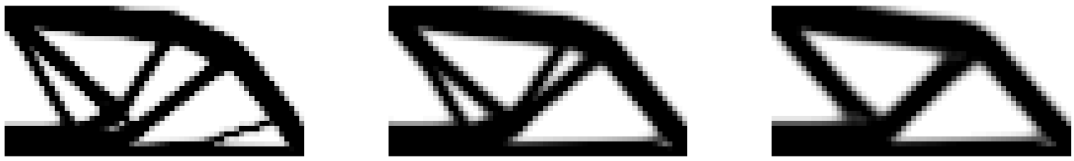
(a) `volfrac`=0.25 (b) `volfrac`=0.5 (c) `volfrac`=0.75

Figure 16: Configurations de poutre pour différentes valeurs de `volfrac`

[3]O. Sigmund, *A 99 line topology optimization code written in Matlab*, Structural and Multidisciplinary Optimization 21(2), 2001, pp. 120-127

(a) `penal`=2 (b) `penal`=3 (c) `penal`=4

Figure 17: Configurations de poutre pour différentes valeurs de `penal`



(a) `rmin`=1.1 (b) `rmin`=1.5 (c) `rmin`=2

Figure 18: Configurations de poutre pour différentes valeurs de `rmin`

La modification de la variable `volfrac` entraine une augmentation de la quantité de matière utilisée. En effet, plus `volfrac` est élevé, plus il faudra un volume conséquent pour satisfaire la contrainte stipulant que le rapport volume de la structure - volume initial doit être égal à `volfrac`.

La variable `penal` intervient dans le calcul de la souplesse sous forme de puissance de x_e. Il peut être remarqué que plus sa valeur augmente, moins il y a de ramifications sur les barres de la structure.

Si `rmin` est faible, la taille minimale des "*tiges*" formant la structure va être petite, et donc il y aura des segments de poutre qui seront très fins comparés aux autres. Inversement, si `rmin` est grand, le nombre de barres diminuera et celles qui seront présentes auront un diamètre plus élevé.

L'article présente aussi une méthode permettant d'ajouter des perçages passifs dans la structure. Ces perçages seront alors considérés par l'algorithme comme des zones où la proportion volumique x_e est toujours égale à x_{min}, et donc où il ne peut y avoir de matière. Pour pouvoir implémenter cette fonction, il suffit juste d'ajouter une petite dizaine de lignes de codes dans le programme (voir l'article pour plus de détails).

Dans la Figure 19 ci-dessous, les 2 versions du programme ont été lancées avec les mêmes paramètres (`nelx`=60, `nely`=30, `volfrac`=0.5, `penal`=3, `rmin`=1.5). Le trou a été placé au centre de la demi-poutre et son rayon est de $\frac{\texttt{nely}}{2}$. On peut voir que la barre qui était au centre dans le cas sans trou a disparu dans le second cas, et que de nouvelles barres sont apparues presque tout autour du trou.

(a) Structure basique (b) Structure avec trou

Figure 19: Formes de la poutre en fonction de la présence ou non d'un trou