Université Paris Saclay

Équilibre d'une chaîne articulée

Auteur :
Antoine Caillebotte
Justine De Sousa

Superviseur:
Benoît Bonnet

March 7, 2021

Sommaire

Introduction

Condidérons le problème consistant à trouver la position d'équilibre statique d'un chaîne formée de barres rigides contenues dans un plan vertical et fixée à ses deux bouts. L'objectif de ce TP est d'écrire un simulateur et un optimiseur qui résoud ce problème.

1 Modélisation du problème *(TP1)*

Dans un premier temps, fixons les notations du problème. On aura:

- (x_i, y_i) les coordonnées de chaque noeud i
- $(x_0, y_0) = (0, 0)$ et $(x_{n_b}, y_{n_b}) = (a, b)$
- n_b le nombre de barres
- L_i la longueur de la i-ème barre
- $n_n = n_b - 1$ le nombre de noeuds (sans compter les extrémités qui sont supposées fixes)

L'extrémité (a, b) pourra varier d'un cas test à l'autre, de même que le nombre de barres n_b et leurs longueurs L_i.

1.1 La fonction objectif

On admettra que la position d'équilibre recherchée est obtenue en minimisant l'énergie potentielle de la chaîne sous des contraintes appropriées. Notons alors $E(x, y)$, l'énergie potentielle de la chaîne qui vaut:

$$E(x, y) = \sum_{i=1}^{n_b} \gamma m_i(x, y) \frac{y_i + y_{i-1}}{2}$$

où $\gamma > 0$ est la constante de gravité et $m_i(x, y)$ la masse de la i-ème barre. Pour simplifier, on supposera

$$\gamma = 1 \quad \text{et} \quad m_i(x, y) = l_i(x, y) = \sqrt{(x_i - x_i - 1)^2 + (y_i - y_i - 1)^2}$$

On obtient alors $E(x, y) = \sum_{i=1}^{n_b} l_i(x, y) \frac{y_i + y_{i-1}}{2}$. On décide de prendre comme variables décrivant la position de la chaîne les coordonnées (x_i, y_i) des noeuds de celle-ci.

1.2 Les contraintes

Il s'agit maintenant de définir les contraintes à respecter. Celles-ci porteront sur les longueurs L_i des barres. La i-ème contrainte s'écrit alors

$$c_i(x, y) := l_i^2 - L_i^2 := (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 - L_i^2 = 0$$

Notons que ces contraintes portent sur le carré des longeurs des barres. Cela permet en effet d'avoir des contraintes différentiables.

1.3 Le problème

Le problème d'optimisation sous contraintes d'égalité ainsi obtenu est alors le suivant: $\begin{cases} \min E(x, y) \\ c_i(x, y) = 0, \quad i = 1, ..., n_b \end{cases}$

Remarquons que sur l'ensemble admissible, on a $E(x, y) = \sum_{i=1}^{n_b} L_i \frac{y_i + y_{i-1}}{2} =: e(x, y)$

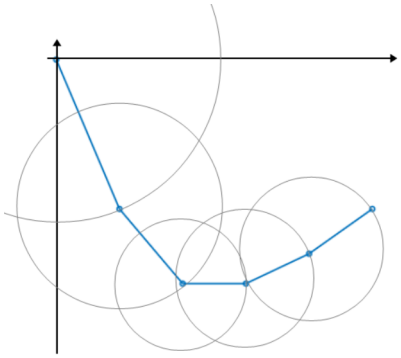
On résoudra alors plutôt le problème avec contraintes linéaires suivant:

$$\begin{cases} \min e(x, y) = \sum_{i=1}^{n_b} L_i \frac{y_i + y_{i-1}}{2} \\ c_i(x, y) = (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 - L_i^2 = 0, \quad i = 1, ..., n_b \end{cases} \tag{P}$$

1.4 Existence d'une solution *(Q1.1)*

Supposons qu'il existe une chaîne admissible, c'est-à-dire une chaîne dont les coordonnées des points vérifient les contraintes c_i : il existe x, y avec $\begin{cases} (x_0, y_0) = (0,0) & (x_{n_b}, y_{n_b}) = (1,-1) \\ c_i(x,y) = 0 & \forall i \in [\![1, n_b]\!] \end{cases}$.
On définit $X_E = \{x, y \mid \forall i \in [\![1, n_b]\!], c_i(x,y) = 0\}$
Montrons que X_E est **compact** :

- On a $X_E = \bigcup_{i=1}^{n} \{(x,y) \mid c_i(x,y) = 0\} = \bigcup_{i=1}^{n} c_i^{-1}(\{0\})$ donc $\boxed{X_E \text{ est fermé}}$



Intuitivement avec l'origine (x_0, y_0) fixé, on comprend que l'on va être borné car chaque points appartient à une boule centré par le points précédent :

- $(x_1, y_1) \in \mathcal{B}((x_0, y_0), L_1)$
- $(x_2, y_2) \in \mathcal{B}((x_1, x_1), L_2)$ soit $(x_2, y_2) \in \mathcal{B}((x_0, y_0), L_1 + L_2)$
- $\forall i \in [\![1, n_b]\!], (x_i, y_i) \in \mathcal{B}\left((x_0, y_0), \sum_{k=1}^{i} L_k\right)$

Donc $\forall (x,y) \in X_E$, x et y sont bornées, soit $\boxed{X_E \text{ est borné}}$.
La fonction objectif e est **continue** sur l'ensemble admissible **non vide** *(par hypothèse)* et **compact**. On a donc **l' existence d'une solution** au problème (P).

1.5 Existence de multiplicateur optimal *(Q1.2)* AFAIRE

Soit $x^\star$ une solution du problème (P). Les contraintes sont **qualifiées** en $x^\star$ car les $\nabla c_i(x)$ sont linéairement indépendants. On a donc l'existence de $\lambda_\star$ tel que

$$\nabla_x l(x_\star, \lambda_\star) = 0.$$

1.6 Implémentation du simulateur

Dans un premier temps, nous implémentons le simulateur. On a alors les variables globales `L`, `A` et `B` et les variables `nn` (n_n), `nb` (n_b). Le simulateur est implémenté à travers la fonction `chs` qui a les entrées suivantes:

xy: le vecteur colonne `xy` $= (x_1, ..., x_{n_n}, y_1, ..., y_{n_n})^\intercal$ à optimiser. Pour simplifier les futurs calculs, on notera `x = [0; xy(1:nn); A]` $((0, x_1, ..., x_{n_n})^\intercal)$ et `y = [0; xy(nn+1:2*nn); B]` $((0, y_1, ..., y_{n_n})^\intercal)$.

lm: le vecteur colonne $\lambda = (\lambda_1, ..., \lambda_{n_b})^\intercal$ des multiplicateurs de lagrange pour les contraintes d'égalité $c_i, i \in E$.

indic: indique au simulateur ce qu'il doit calculer. Les sorties dépendent donc de la valeur de l'entrée `indic`:

indic = 1: tracé de la chaîne

indic = 2: calcul de e et c où

- e est l'énergie potentielle de la chaine, fonction objectif de notre problème. On la calcule selon la formule (1.3). Ce qui donne e $= \langle L, Y_+ + Y_- \rangle / 2$ où $Y_+ = (y_1, ..., y_{n_n}, B)$ et $Y_- = (0, y_1, ..., y_{n_n})$.
- c est le vecteur colonne des contraintes de notre problème en `xy`. On le calcule selon la formule (1.2). Ce qui donne c $= ||X_+ - X_-||^2 + ||Y_+ - Y_-||^2 - ||L||^2$ où $X_+ = (x_1, ..., x_{n_n}, A)$ et $X_- = (0, x_1, ..., x_{n_n})$

indic = 4: calcul de e, c, g et a où :

- g est le gradient de e, vecteur colonne de taille $2 * n_n$. A partir de (1.3), on en déduit que

$$e(x,y) = \underbrace{\frac{0 + L_1}{2} y_0}_{=0} + \frac{L_1 + L_2}{2} y_1 + ... + \frac{L_{n_b-1} + L_{n_b}}{2} y_{n_b-1} + \frac{L_{n_b}}{2} y_{n_b}$$

Et donc,

$$\forall i \in 1, ..., n_n \quad \frac{\partial e(x,y)}{\partial x_i} = 0 \quad \text{et} \quad \frac{\partial e(x,y)}{\partial y_i} = \frac{L_i + L_{i+1}}{2}$$

avec $L_{n_b+1} = 0$.

Ce qui donne g $= \begin{pmatrix} 0 \\ \vdots \\ 0 \\ L_- + L_+ \end{pmatrix}$ où $L_- = (L_1, ..., L_{n_b+1})^\intercal$ et $L_+ = (L_2, ..., L_{n_b})^\intercal$

- a est la jacobienne des contraintes, matrice de taille $n_b \times 2n_n$. D'après (1.2), on a :

$$\frac{\partial c_i(x,y)}{\partial x_k} = \begin{cases} 2(x_{i-1} - x_i) & \text{si } k = i-1 \\ 2(x_i - x_{i-1}) & \text{si } k = i \\ 0 & \text{sinon} \end{cases} \quad \text{et} \quad \frac{\partial c_i(x,y)}{\partial y_k} = \begin{cases} 2(y_{i-1} - y_i) & \text{si } k = i-1 \\ 2(y_i - y_{i-1}) & \text{si } k = i \\ 0 & \text{sinon} \end{cases}$$

Ainsi, pour $n_b = 5$, on a :
$c'(x,y) =$

$$2 \times \begin{pmatrix} x_1 - x_0 & 0 & 0 & 0 & 0 & y_1 - y_0 & 0 & 0 & 0 & 0 \\ x_1 - x_2 & x_2 - x_1 & 0 & 0 & 0 & y_1 - y_2 & y_2 - y_1 & 0 & 0 & 0 \\ 0 & x_2 - x_3 & x_3 - x_2 & 0 & 0 & 0 & y_2 - y_3 & y_3 - y_2 & 0 & 0 \\ 0 & 0 & x_3 - x_4 & x_4 - x_3 & 0 & 0 & 0 & y_3 - y_4 & y_4 - y_3 & 0 \\ 0 & 0 & 0 & x_4 - A & A - x_4 & 0 & 0 & 0 & y_4 - B & B - y_4 \end{pmatrix}$$

indic = 5: calcul de `h1`, le hessien du lagrangien en (`xy`,`lm`)

$$\nabla^2_{xx} l(x, \lambda) = \underbrace{\nabla^2 e(x)}_{=0} + \sum_{i=1}^{n_b} \lambda_i \nabla^2 c_i(x)$$

On a, pour $n_b = 5$

$\nabla^2 c_1(x,y) =$

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$\nabla^2 c_2(x,y) =$

$$\begin{pmatrix} 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$\nabla^2 c_3(x,y) =$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\nabla^2 c_4(x,y) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 \end{pmatrix} \quad \text{Et } \nabla^2 c_5(x,y) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

D'où $\texttt{hl} = \sum_{i=1}^{n_b} \lambda_i \nabla^2 c_i(x) =$

$$2 \times \begin{pmatrix} \lambda_1+\lambda_2 & -\lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\lambda_2 & \lambda_2+\lambda_3 & -\lambda_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\lambda_3 & \lambda_3+\lambda_4 & -\lambda_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\lambda_4 & \lambda_4+\lambda_5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_1+\lambda_2 & -\lambda_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\lambda_2 & \lambda_2+\lambda_3 & -\lambda_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\lambda_3 & \lambda_3+\lambda_4 & -\lambda_4 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_4 & \lambda_4+\lambda_5 \end{pmatrix}$$

1.7 Validation des calculs numériques

Nous vérifions ensuite l'exactitude des calculs numériques effectués. Premièrement, nous vérifions sur des valeurs connues que le calcul de e, c et g est correct. Les résultats trouvés sont les suivants:

```
e = -2.2800      c =                g =
                      0.550000          0.00000
                      0.040000          0.00000
                     -0.050000          0.00000
                      0.040000          0.00000
                     -0.120000          0.00000
                                        0.60000
                                        0.40000
                                        0.25000
                                        0.35000
```

Ensuite, nous comparons le calcul du gradient de e et un calcul par différences finies. La fonction `verifierGradient` affiche alors:

```
-----------------------------------------------
 i    pas       f'(i)         DF         erreur
-----------------------------------------------
 1 1.49e-08 0.00000e+00 0.00000e+00 0.00000e+00
 2 1.49e-08 0.00000e+00 0.00000e+00 0.00000e+00
 3 1.49e-08 0.00000e+00 0.00000e+00 0.00000e+00
 4 1.49e-08 0.00000e+00 0.00000e+00 0.00000e+00
 5 1.49e-08 6.00000e-01 6.00000e-01 9.93411e-09
 6 1.49e-08 4.00000e-01 4.00000e-01 1.49012e-08
 7 1.49e-08 2.50000e-01 2.50000e-01 5.96046e-08
 8 1.49e-08 3.50000e-01 3.50000e-01 1.70299e-08
-----------------------------------------------
```

Nous calculons aussi la jacobienne des contraintes et vérifions le résultat à la main:

```
a =
    0.4000  0.0000  0.0000  0.0000  -2.0000  0.0000  0.0000  0.0000
   -0.4000  0.4000  0.0000  0.0000   1.0000 -1.0000  0.0000  0.0000
    0.0000 -0.4000  0.4000  0.0000   0.0000  0.0000  0.0000  0.0000
    0.0000  0.0000 -0.4000  0.4000   0.0000  0.0000 -0.4000  0.4000
    0.0000  0.0000  0.0000 -0.4000   0.0000  0.0000  0.0000 -0.6000
```

Enfin, nous calculons le hessien du lagrangien:

```
hl =
    1.86006   -0.84460    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
   -0.84460    1.88250   -1.03791    0.00000    0.00000    0.00000    0.00000    0.00000
    0.00000   -1.03791    2.26912   -1.23121    0.00000    0.00000    0.00000    0.00000
    0.00000    0.00000   -1.23121    2.98599    0.00000    0.00000    0.00000    0.00000
    0.00000    0.00000    0.00000    0.00000    1.86006   -0.84460    0.00000    0.00000
    0.00000    0.00000    0.00000    0.00000   -0.84460    1.88250   -1.03791    0.00000
    0.00000    0.00000    0.00000    0.00000    0.00000   -1.03791    2.26912   -1.23121
    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000   -1.23121    2.98599
```

2 Méthode de résolution du problème *(TP2)*

Rappelons le problème d'optimisation sous contraintes qu'on cherche à résoudre:

$$\boxed{\begin{cases}\min e(x,y)\\ c_i(x,y)=0, \quad \text{pour } i=1,...,n_b\end{cases}} \tag{P}$$

On écrit les conditions d'optimalité sur le lagrangien:

$$\begin{cases}\nabla_x l(x_\star,\lambda_\star)=0\\ c(x_\star)=0\end{cases} \quad \Leftrightarrow \quad \boxed{\begin{cases}\nabla f(x_\star)+c'(x_\star)^\intercal\lambda_\star=0\\ c(x_\star)=0\end{cases}} \tag{L}$$

On résoudra alors plutôt ce problème plus simple qui consiste à trouver un point stationnaire. On appliquera un algorithme de Newton pour la recherche de zéro à la fonction $F:\mathbb{R}^N\mapsto\mathbb{R}^N$ suivante avec $z=(x,\lambda)\in\mathbb{R}^n\times\mathbb{R}^m$:

$$F(z)=\begin{pmatrix}\nabla f(x)+c'(x)^\intercal\lambda\\ c(x)\end{pmatrix} \quad \text{dont la dérivé s'écrit} \quad F'(z)=\begin{pmatrix}\nabla^2_{xx}l(x,\lambda) & c'(x)^\intercal\\ c'(x) & 0\end{pmatrix}$$

Ainsi, l'algorithme de Newton consiste à calculer à chaque itération $p_k=\begin{pmatrix}d_k\\ \mu_k\end{pmatrix}$ tel que

$$F'(z_k)p_k=-F(z_k) \quad \Leftrightarrow \quad \begin{pmatrix}\nabla^2_{xx}l(x_k,\lambda_k) & c'(x_k)^\intercal\\ c'(x_k) & 0\end{pmatrix}\begin{pmatrix}d_k\\ \mu_k\end{pmatrix}=-\begin{pmatrix}\nabla f(x_k)+c'(x_k)^\intercal\lambda_k\\ c(x_k)\end{pmatrix}$$

Puis le nouvel itéré est :

$$z_{k+1}=z_k+p_k \quad \Leftrightarrow \quad \begin{pmatrix}x_{k+1}\\ \lambda_{k+1}\end{pmatrix}=\begin{pmatrix}x_k\\ \lambda_k\end{pmatrix}+\begin{pmatrix}d_k\\ \mu_k\end{pmatrix}$$

En introduisant $\lambda_k^{PQ}=\lambda_k+\mu_k$, on peut simplifier chaque itération. En effet, on obtient $\begin{pmatrix}d_k\\ \lambda_k^{PQ}\end{pmatrix}$ tel que

$$\boxed{\begin{pmatrix}\nabla^2_{xx}l(x_k,\lambda_k) & c'(x_k)^\intercal\\ c'(x_k) & 0\end{pmatrix}\begin{pmatrix}d_k\\ \lambda_k^{PQ}\end{pmatrix}=-\begin{pmatrix}\nabla f(x_k)\\ c(x_k)\end{pmatrix}}$$

Et le nouvel itéré vaut alors :

$$\boxed{\begin{pmatrix}x_{k+1}\\ \lambda_{k+1}\end{pmatrix}=\begin{pmatrix}x_k+d_k\\ \lambda_k^{PQ}\end{pmatrix}}$$

2.1 Implémentation de l'optimiseur

Nous implémentons l'optimiseur `sqp.m`. Celui-ci prend les arguments d'entrée suivants:

simul : le simulateur, à savoir ici la fonction `chs.m`.

x : le vecteur à optimiser

lm : les multiplicateurs de lagrange.

2.1.1 Calcul du premier multiplicateur *(Q2.2)*

Il se peut aussi qu'ils ne soient pas donnés. Dans ce cas, ils seront calculés en utilisant une **condition nécessaire d'ordre 1** :
Soit x un point stationnaire de notre problème, alors il existe λ tel que

$$\nabla_x l(x,\lambda) = 0 \quad \Leftrightarrow \quad \nabla f(x) + c'(x)^\intercal \lambda = 0$$

Cette équation n'admet pas de solution car $c'(x)$ n'est pas carrée et donc pas inversible. On veut donc résoudre "au mieux" cette équation. Cela peut se faire par la méthode des **moindres carrés** qui consiste à résoudre le problème d'optimisation

$$\min_{\lambda \in \mathbb{R}^n_+} \|\nabla f(x) + c'(x)^\intercal \lambda\|_2^2$$

Cela revient à écrire `c'(x)'\∇f'(x)` sur MATLAB .

---

options: une structure donnant les différentes options de l'optimiseur:

- `options.tol(1)` et `options.tol(2)` les conditions d'arrêt de l'algorithme qui portent sur la norme du **gradient du lagrangien** et la **norme des contraintes**.
- `options.maxit` donne le nombre d'itérations maximum de l'algorithme.

Les arguments de sortie sont les suivants:

x: Le vecteur optimisé

lm: les multiplicateurs lagrangiens

info: les informations sur ce que l'algorithme a réussi à calculer:

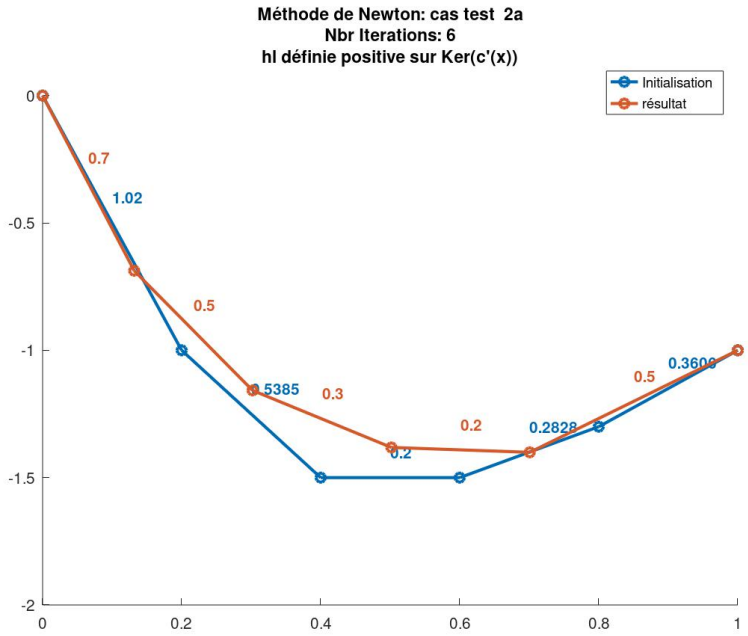
- `info.status == 0`: l'algorithme a tourné normalement et le seuil d'optimisation a été atteint
- `info.status == 1`: inconsistance des arguments d'entrée
- `info.status == 2`: l'algorithme a tourné jusqu'au nombre maximum d'itérations autorisées
- `info.niter`: le nombre d'itération effectuées

2.2 Etude de plusieurs conditions initiales

Dans tous les cas-test effectués à la subsection 2.2 , nous prendrons:

```
(A,B) = (1,-1)
L = (0.7, 0.5, 0.3, 0.2, 0.5)
```

2.2.1 Cas-test 2a *(Q2.1)*



Conditions initiales:

```
x = (0.2, 0.4, 0.6, 0.8)'
y = (-1 ,-1.5,-1.5,-1.3)'
```

Solution:

```
x = ( 0.13168, 0.30198, 0.50170, 0.70078)'
y = (-0.68750,-1.15761,-1.38147,-1.40059)'
```

Convergence de l'algorithme: L'algorithme converge alors en 6 itérations:

```
iter    |gl|        |ce|        |x|        |lm|       alpha       phi         Q
-------------------------------------------------------------------------------
   0 1.0471e-01 5.5000e-01 1.5000e+00 8.7739e-01 1.0000e+00 5.1380e-01 5.5000e-01
   1 3.0020e-01 8.5250e-02 1.5171e+00 7.6378e-01 1.0000e+00 3.5983e-01 9.9238e-01
   2 7.2823e-02 1.6906e-02 1.4289e+00 8.7199e-01 1.0000e+00 3.5272e-01 8.0809e-01
   3 1.7632e-02 1.8823e-03 1.4043e+00 9.2596e-01 1.0000e+00 3.5250e-01 3.3247e+00
   4 3.3904e-04 7.2540e-05 1.4006e+00 9.2604e-01 1.0000e+00 3.5250e-01 1.0906e+00
   5 4.8691e-07 6.5661e-08 1.4006e+00 9.2613e-01 1.0000e+00 3.5250e-01 4.2359e+00
   6 3.2027e-13 4.5353e-14 1.4006e+00 9.2613e-01 1.0000e+00 3.5250e-01 1.3509e+00
```

Vitesse de convergence de z_k vers $z_\star$ *(Q2.1)* Comme F est différentiable et pour z_k dans un voisinage de $z_\star$ *(solution de $F(z) = 0$)*, on a $F(z_k) = \underbrace{F(z_\star)}_{=0} + F'(z_\star)(z_k - z_\star)$ et donc $F(z_k) \sim z_k - z_\star$. On peut donc étudier le comportement de $\|F(z_k)\|_\infty$ pour connaître la convergence de l'algorithme de Newton **sans pour autant connaître la solution** $z_\star$. On souhaite donc savoir si on a bien une convergence quadratique de l'algorithme vers la solution, c'est-à-dire

$$\|F(z_{k+1})\|_\infty \leqslant C \|F(z_k)\|_\infty^2 \tag{1}$$

Or

$$\|F(z_k)\|_\infty = \max(\|\nabla_x l(x_k, \lambda_k)\|_\infty, \|c(x_k)\|_\infty)$$

Dans la dernière colonne du tableau, on observe la valeur du quotient $Q = \dfrac{\|F(z_{k+1})\|}{\|F(z_k)\|^2}$. On peut donc prendre comme constante $C = 4.5$ qui nous donne l'inégalité de convergence quadratique (1).

On peut aussi observer directement $\|gl\|$ et $\|ce\|$ dont la norme est divisée par 2 à chaque itération.

Type de point stationnaire: Etudions les conditions nécessaires d'ordre 2 de notre solution. On calcule le gradient du lagrangien et les contraintes en ce point.

```
Grdl(x,lm) =-9.65894e-15
            +3.20272e-13
            -9.16489e-14
            -2.16271e-13
            -5.32907e-15
            +5.96190e-14
            -2.78000e-13
            +2.32203e-13
```

```
c(x) =
          +4.99600e-15
          +6.93889e-15
          +4.29101e-14
          +2.78111e-14
          +4.53526e-14
```

On a donc $\nabla l(x,\lambda) = 0$ et $c(x) = 0$ (à la précision machine). (x,λ) est donc bien un **point stationnaire**, calculons le hessien du lagrangien sur le noyau de $c'(x)$. La fonction `null()` de MATLAB permet de calculer le noyau de $c'(x)$.

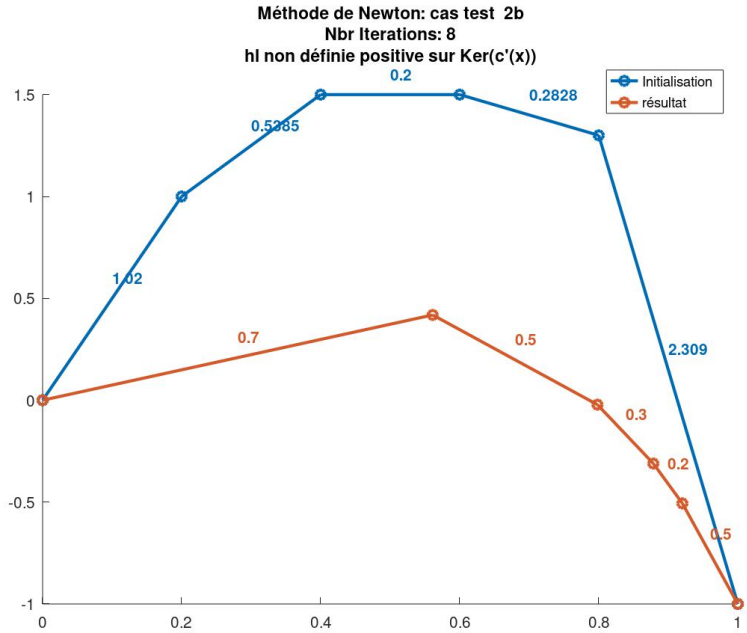
```
null(a) =
  -0.296944   -0.356157   -0.838459
   0.803736   -0.247550   -0.170382
   0.203381    0.464189   -0.317237
   0.207133    0.378816   -0.291559
  -0.056882   -0.068225   -0.160613
   0.341817   -0.028884    0.081384
  -0.193787    0.606091   -0.049632
  -0.154717   -0.282955    0.217778
```

On calcule ensuite $< \nabla^2_{xx} l(x,\lambda).z_i | z_i >$ pour tout élément $z_i \neq 0$ du noyau. On obtient:

```
<Grd^2l(x,lm).z_1|z_1> = 2.979470
<Grd^2l(x,lm).z_2|z_2> = 2.533538
<Grd^2l(x,lm).z_3|z_3> = 2.316885
```

On a donc $\langle \nabla^2_{xx} l(x,\lambda).z_i | z_i \rangle > 0$ pour tout z_i dans le noyau de $c'(x)$, ce qui signifie que $\nabla^2_{xx} l(x,\lambda)$ est **défini positif sur le noyau** de $c'(x)$ et donc que notre solution est un **minimum local** strict.

2.2.2 Cas-test 2.b



Conditions initiales:

```
x = (0.2, 0.4, 0.6, 0.8)'
y = (1.0, 1.5, 1.5, 1.3)'
```

Solution:

```
x = ( 0.56113, 0.79812, 0.87854, 0.92035)'
y = ( 0.41849,-0.02179,-0.31080,-0.50639)'
```

Convergence de l'algorithme: L'algorithme converge en 8 itérations:

```
iter    |gl|        |ce|        |x|        |lm|       alpha       phi         Q
-------------------------------------------------------------------------------
   0 1.0947e-01 5.0800e+00 1.5000e+00 4.9918e-01 1.0000e+00 1.3410e+01 5.0800e+00
   1 7.1927e-01 1.2716e+00 1.1401e+00 1.0669e+00 1.0000e+00 1.2181e+00 4.9275e-02
```

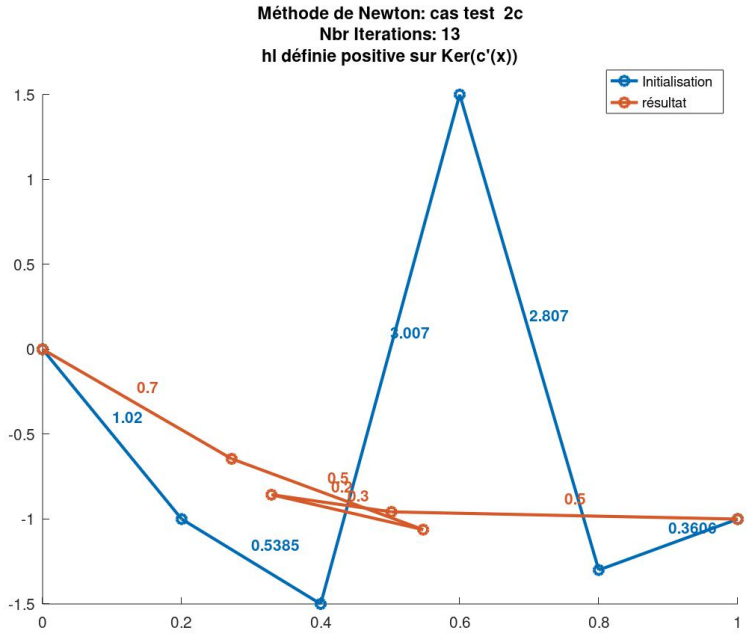
```
 2 2.7989e-01 5.2561e-01 1.0669e+00 1.0005e+00 1.0000e+00 5.6387e-01 3.2505e-01
 3 1.4653e-01 3.0701e-01 8.9180e-01 1.4460e+00 1.0000e+00 4.0667e-01 1.1113e+00
 4 2.3183e-01 1.5877e-01 8.7834e-01 2.4836e+00 1.0000e+00 3.7470e-01 2.4597e+00
 5 3.4823e-02 3.5448e-02 9.2675e-01 2.6189e+00 1.0000e+00 3.5340e-01 6.5956e-01
 6 3.9590e-03 1.5451e-03 9.2029e-01 2.7438e+00 1.0000e+00 3.5250e-01 3.1506e+00
 7 2.2552e-05 4.8490e-06 9.2035e-01 2.7562e+00 1.0000e+00 3.5250e-01 1.4388e+00
 8 2.9583e-10 6.0701e-11 9.2035e-01 2.7562e+00 1.0000e+00 3.5250e-01 5.8168e-01
-------------------------------------------------------------------------------
```

On remarque une **convergence quadratique** de l'algorithme *(grâce à $\|g1\|$ et $\|ce\|$)*.

Type de point stationnaire: On a $\nabla l(x,\lambda) = 0$ et $c(x) = 0$ donc (x,λ) est un point stationnaire. On calcule le hessien du lagrangien et le noyau de $c'(x)$ (`N`). La projection de la hessienne sur `N` s'écrit `N'hlN`. On regarde ensuite les valeurs propres de cette matrice: `eig(N'hlN) = (-14.2644, -5.0431, -1.0438)`

On a $\nabla^2_{xx} l(x,\lambda) < 0$ donc $\nabla^2_{xx}(-l)(x,\lambda) > 0$. Or $-l(x,\lambda) = -f + \langle\lambda| - c(x)\rangle$ est le lagrangien du problème suivant : $\begin{cases}\min -f \\ c(x) = 0\end{cases}$. Donc x est un minimum local de $-f$ et donc x **est un maximum local de** f

2.2.3 Cas-test 2.c



Conditions initiales:

```
x = ( 0.2, 0.4, 0.6, 0.8)'
y = (-1.0,-1.5, 1.5,-1.3)'
```

Solution:

```
x = ( 0.27197, 0.54751, 0.32921, 0.50181)'
y = (-0.64500,-1.06223,-0.85645,-0.95750)'
```

Convergence de l'algorithme: L'algorithme converge en 13 itérations:

```
iter    |gl|        |ce|        |x|        |lm|       alpha       phi          Q
-------------------------------------------------------------------------------
 0 2.1880e-01 8.9500e+00 1.5000e+00 5.5533e-01 1.0000e+00 7.1296e+01 8.9500e+00
 1 7.9949e+00 2.9997e+01 5.7814e+00 2.1782e+00 1.0000e+00 8.9366e+02 3.7448e-01
 2 1.9500e+00 7.7894e+00 2.3971e+00 1.0988e+00 1.0000e+00 5.9432e+01 8.6568e-03
 3 4.2791e-01 2.4880e+00 1.2279e+00 1.0924e+00 1.0000e+00 5.6698e+00 4.1005e-02
 4 3.8369e-01 1.8083e+00 1.4472e+00 1.2740e+00 1.0000e+00 2.8559e+00 2.9213e-01
 5 6.5083e-02 9.2896e-01 1.2156e+00 1.0586e+00 1.0000e+00 9.8552e-01 2.8410e-01
 6 1.1397e-01 2.2913e-01 1.1650e+00 1.1262e+00 1.0000e+00 3.8907e-01 2.6552e-01
 7 3.5081e-01 1.0667e-01 1.1010e+00 1.1952e+00 1.0000e+00 3.6275e-01 6.6819e+00
```

```
 8 1.2601e-01 4.7550e-02 1.0805e+00 1.2123e+00 1.0000e+00 3.5426e-01 1.0239e+00
 9 8.6632e-02 1.0496e-02 1.0779e+00 1.5546e+00 1.0000e+00 3.5256e-01 5.4560e+00
10 2.2125e-02 8.5912e-04 1.0627e+00 1.9427e+00 1.0000e+00 3.5250e-01 2.9479e+00
11 6.6840e-04 2.2780e-05 1.0623e+00 2.0256e+00 1.0000e+00 3.5250e-01 1.3655e+00
12 3.0782e-07 1.1248e-08 1.0622e+00 2.0274e+00 1.0000e+00 3.5250e-01 6.8900e-01
13 1.3672e-13 7.8271e-15 1.0622e+00 2.0274e+00 1.0000e+00 3.5250e-01 1.4430e+00
-------------------------------------------------------------------------------
```

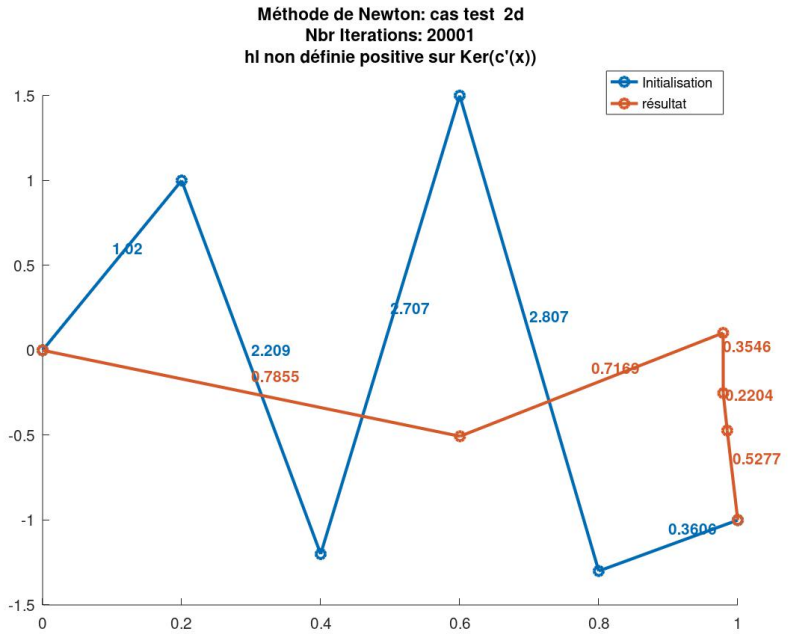
On remarque une **convergence quadratique** de l'algorithme.

Type de point stationnaire: (x, λ) est un point stationnaire et

```
eig(N'hlN) =  -3.6139
               6.6495
               5.2384
```

Donc $\nabla^2_{xx} l(x, \lambda)$ a des valeurs propres positives et négatives sur le noyau de $c'(x)$. Notre solution est donc un **point selle**.

2.2.4 Cas-test 2.d



Conditions initiales:

```
x = ( 0.2, 0.4, 0.6, 0.8)'
y = ( 1.0,-1.2, 1.5,-1.3)'
```

Solution:

```
x = ( 0.60041, 0.97896, 0.97907, 0.98480)'
y = (-0.50644, 0.10240,-0.25221,-0.47252)'
```

Convergence de l'algorithme: L'algorithme ne converge pas à priori :

```
iter     |gl|        |ce|        |x|        |lm|       alpha       phi          Q
-------------------------------------------------------------------------------
19995 3.3370e-01 2.1984e-01 1.4057e+00 2.4382e+00 1.0000e+00 3.8300e-01 2.3980e+00
19996 4.2885e-01 2.8252e-01 9.7509e-01 2.3364e+00 1.0000e+00 4.0288e-01 3.8511e+00
19997 2.8725e-01 1.8924e-01 1.3236e+00 2.4197e+00 1.0000e+00 3.7510e-01 1.5619e+00
19998 1.1172e+00 7.3601e-01 8.3790e-01 2.2554e+00 1.0000e+00 6.9442e-01 1.3540e+01
19999 3.6779e-01 2.4229e-01 1.0125e+00 2.3497e+00 1.0000e+00 3.8955e-01 2.9466e-01
20000 3.4158e-01 2.2503e-01 1.4161e+00 2.4405e+00 1.0000e+00 3.8446e-01 2.5252e+00
-------------------------------------------------------------------------------
```

On obtient les résultats suivants:

```
Grdl(x,lm) = -6.32517e-02
             +1.03268e-01
             +2.66064e-02
             +1.30881e-02
             +4.00721e-01
             -2.05931e-01
             -1.03228e-02
             -3.19161e-03
```

```
c(x) =   +1.26977e-01
         +2.63988e-01
         +3.57522e-02
         +8.56828e-03
         +2.84680e-02
```

Donc la solution trouvée n'est **pas un point stationnaire**. Ceci peut s'expliquer par le fait que la condition initiale est trop éloignée d'un point stationnaire.

3 Globalisation par recherche linéaire *(TP3)*

3.1 Motivation *(Q3.2)*

On a vu que l'algorithme de Newton tel qu'utilisé précedemment risque de ne pas converger lorsqu'on part d'un point trop éloigné de la solution. L'objectif de cette section sera alors de rendre l'algorithme global. Pour cela, on va chercher un pas α optimal tel que l'itéré de l'algorithme de Newton sera

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k + \alpha_k d_k \\ \lambda_k + \alpha_k \mu_k \end{pmatrix}$$

Où α_k sera défini par la *règle d'Armijo*. C'est-à-dire que $\alpha = 2^{-i_k}$ où i_k est le plus petit entier tel que

$$\varphi(z_k + \alpha_k p_k) \leqslant \varphi(z_k) + \omega \alpha_k \varphi'(z_k).p_k$$

On commence avec $\alpha_k = 1$ car dans ce cas la condition d'Armijo est vérifiée :

$$\varphi(z_k + p_k) \leqslant \varphi(z_k) + \omega \varphi'(z_k) p_k = \varphi(z_k) - \omega \left\| F(z) \right\|_2^2 \leqslant \varphi(z_k)$$

Donc $z_k + p_k$ améliore le point courant.

Montrons qu'il existe $\alpha > 0$ tel que α vérifie l'inégalité d'Armijo :

$$\frac{\varphi(z_k + t d_k) - \varphi(z_k)}{t} \underset{t \longrightarrow \infty}{\rightarrow} - \left\| F(z_k) \right\|_2^2$$

Or, $\omega \leqslant 1$ donc il existe t_0 tel que

$$\frac{\varphi(z_k + t_0 d_k) - \varphi(z_k)}{t_0} \leqslant -\omega \left\| F(z_k) \right\|_2^2$$

Ainsi, pour $\alpha \leqslant t_0$, la condition d'Armijo est vérifiée.

3.2 Direction de descente *(Q3.1)*

En effet, p_k est une **direction de descente** en z_k de la fonction φ puisque:

$$\varphi'(z_k) p_k = (\nabla F(z_k)^\intercal F(z_k))^\intercal . p_k = F(z_k)^\intercal \nabla F(z_k) . p_k = -F(z_k)^\intercal F(z_k) = - \left\| F(z_k) \right\|^2 \leqslant 0$$

3.3 L'algorithme

Pour l'écriture de l'algorithme, nous avons besoin d'exprimer λ_{k+1} en fonction λ_k et λ^{PQ}.
On sait que $\lambda^{PQ} = \lambda_k + \mu_k$ donc $\mu_k = \lambda^{PQ} - \lambda_k$.
Or : $\lambda_{k+1} = \lambda_k + \alpha \mu_k = \lambda_k + \alpha(\lambda^{PQ} - \lambda_k) = (1 - \alpha)\lambda_k + \alpha \lambda^{PQ}$

De plus, nous avons besoin de calculer φ':
Soit $h > 0$, par Taylor on a :

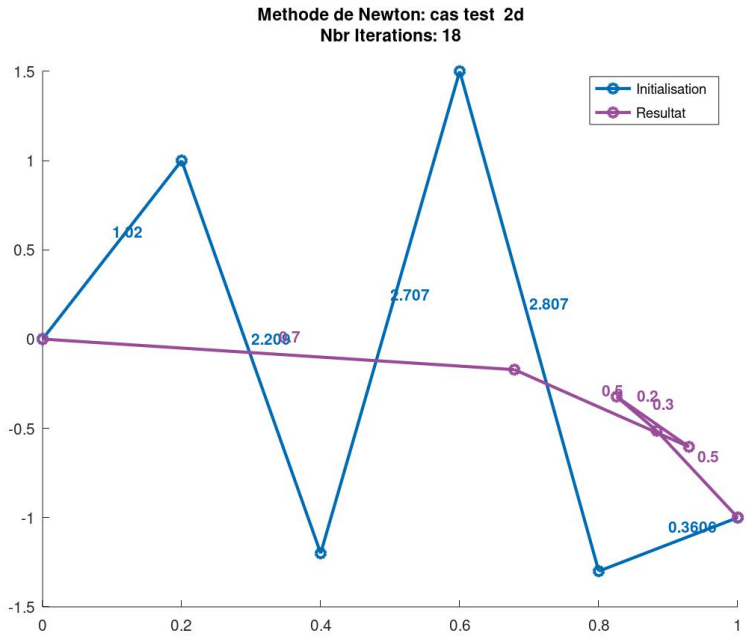
$$\begin{aligned} \left\| F(z+h) \right\|_2^2 &= ||F(z) + \nabla F(z).h + o(h)||_2^2 \\ &= \left\| F(z) \right\|_2^2 + 2\langle F(z), \nabla F(z).h + o(h) \rangle + \left\| \nabla F(z).h + o(h) \right\|_2^2 \\ &= \left\| F(z) \right\|_2^2 + 2\langle F(z), \nabla F(z).h \rangle + o\left(\left\| h \right\|_2^2 \right) \\ &= \left\| F(z) \right\|_2^2 + 2F(z)^T \nabla F(z).h + o\left(\left\| h \right\|_2^2 \right) \\ &= \left\| F(z) \right\|_2^2 + 2(\nabla F(z) F(z))^T.h + o\left(\left\| h \right\|_2^2 \right) \end{aligned}$$

Donc : $\varphi(z+h) - \varphi(z) = (\nabla F(z) F(z))^T.h + o\left(\left\| h \right\|_2^2 \right)$ Ainsi :

$$\boxed{\varphi'(z).h = \nabla F(z)^T F(z).h}$$

3.4 Etude de plusieurs conditions initiales

3.4.1 Cas test 2d



Convergence de l'algorithme: L'algorithme converge en 18 itérations:

iter	|gl|	|ce|	|x|	|lm|	alpha	phi	Q
0	2.6828e-01	7.8400e+00	1.5000e+00	5.4647e-01	1.0000e+00	6.7877e+01	7.8400e+00
1	2.4956e-01	7.7026e+00	1.5261e+00	6.3918e-01	3.1250e-02	6.5760e+01	1.2532e-01
2	2.9238e-01	2.0650e+00	9.7172e-01	1.0529e+00	1.0000e+00	5.6022e+00	3.4805e-02
3	4.4940e-01	1.5837e+00	7.4264e-01	1.2654e+00	5.0000e-01	3.0113e+00	3.7139e-01
4	5.5588e-01	1.6550e+00	8.6245e-01	1.7665e+00	5.0000e-01	2.6632e+00	6.5986e-01
5	3.2658e-01	1.6276e+00	7.2371e-01	2.4858e+00	2.5000e-01	2.1779e+00	5.9425e-01
6	3.4954e-01	1.5776e+00	7.5310e-01	6.0742e-01	3.1250e-02	2.1239e+00	5.9551e-01
7	7.6374e-01	1.4987e+00	1.0802e+00	1.0789e+00	2.5000e-01	2.0865e+00	6.0218e-01
8	6.7127e-01	8.8351e-01	1.2647e+00	6.4319e-01	1.0000e+00	9.2421e-01	3.9335e-01
9	6.3062e-01	3.8424e-01	8.5101e-01	1.3333e+00	1.0000e+00	4.4200e-01	8.0786e-01
10	2.1263e-01	1.4549e-01	9.6388e-01	1.5428e+00	1.0000e+00	5.2545e-02	5.3467e-01
11	8.1372e-02	8.1474e-02	9.3738e-01	1.6802e+00	5.0000e-01	1.5191e-02	1.8021e+00
12	8.7556e-02	4.4406e-02	9.3242e-01	2.0797e+00	5.0000e-01	9.0460e-03	1.3190e+01
13	7.5649e-02	2.3725e-02	9.3140e-01	2.4900e+00	5.0000e-01	5.7633e-03	9.8680e+00
14	6.9040e-02	3.4034e-03	9.3044e-01	3.2185e+00	1.0000e+00	4.5581e-03	1.2064e+01
15	3.1844e-03	8.4017e-05	9.2989e-01	3.5214e+00	1.0000e+00	1.1327e-05	6.6806e-01
16	2.0759e-06	2.8775e-08	9.2981e-01	3.5331e+00	1.0000e+00	4.8686e-12	2.0472e-01
17	1.9475e-12	3.1086e-14	9.2981e-01	3.5331e+00	1.0000e+00	3.8658e-24	4.5190e-01

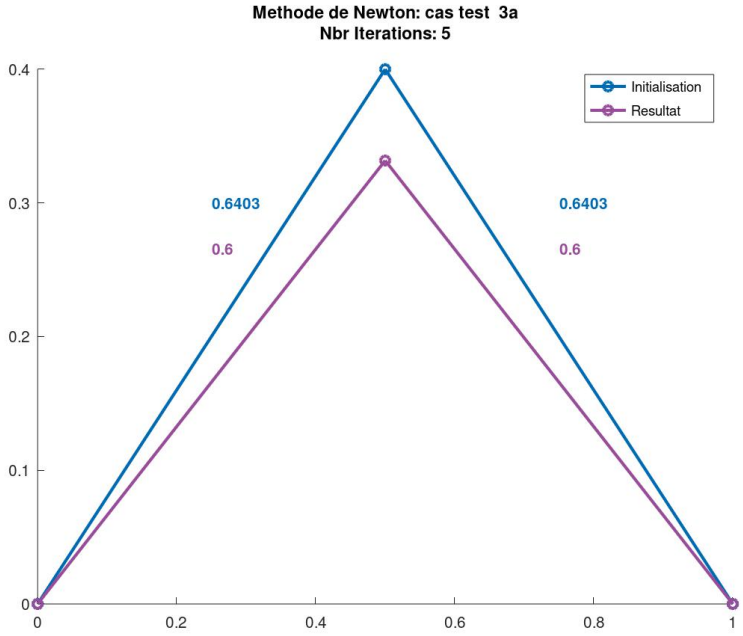
On remarque une **convergence quadratique** de l'algorithme.

Type de point stationnaire: (x, λ) est un point stationnaire et

```
eig(N'hlN) =  4.9446
             -1.8334
             -15.0153
```

Donc $\nabla^2_{xx} l(x, \lambda)$ a des valeurs propres positives et négatives sur le noyau de $c'(x)$. Notre solution est donc un **point selle**.

3.4.2 Cas test 3a *A COMPLETER*



Convergence de l'algorithme: L'algorithme converge en 5 itérations:

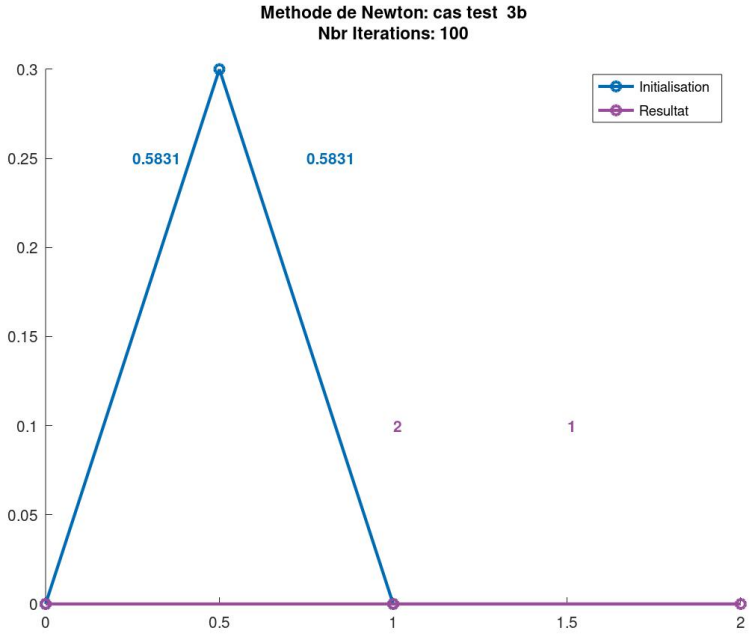
```
iter    |gl|        |ce|         |x|        |lm|       alpha       phi          Q
-------------------------------------------------------------------------------------
   0 0.0000e+00 5.0000e-02 5.0000e-01 3.7500e-01 1.0000e+00 2.5000e-03 5.0000e-02
   1 1.4648e-02 3.9062e-03 5.0000e-01 4.3359e-01 1.0000e+00 1.2255e-04 5.8594e+00
   2 4.2327e-04 3.3490e-05 5.0000e-01 4.5188e-01 1.0000e+00 9.0702e-08 1.9726e+00
   3 7.8299e-08 2.5482e-09 5.0000e-01 4.5227e-01 1.0000e+00 3.0719e-15 4.3703e-01
   4 9.9920e-16 0.0000e+00 5.0000e-01 4.5227e-01 1.0000e+00 4.9920e-31 1.6298e-01
-------------------------------------------------------------------------------------
```

On remarque une **convergence quadratique** de l'algorithme.

Type de point stationnaire: (x, λ) est un point stationnaire et `eig(N'hlN) =`

Donc ...

3.4.3 Cas test 3b



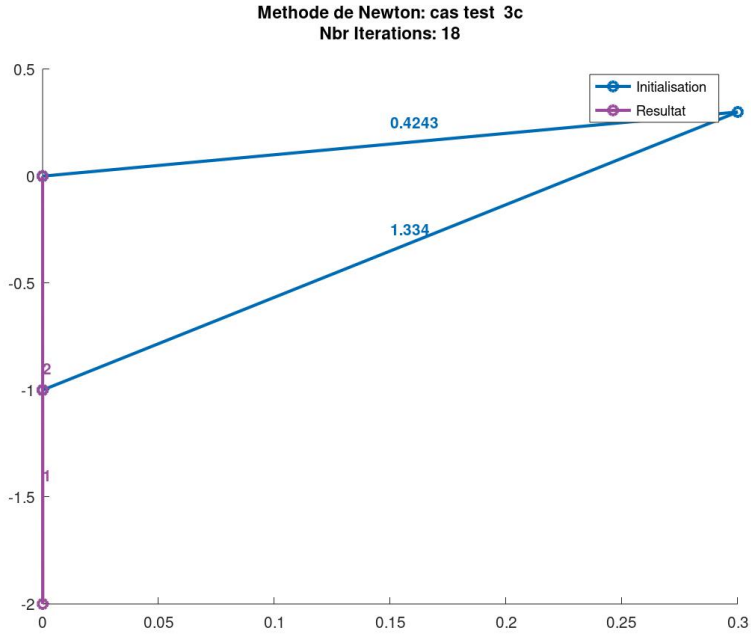

```
-------------------------------------------------------------------------------
iter    |gl|        |ce|        |ci|        |x|       |lme|       |lmi|        alpha
-------------------------------------------------------------------------------
   0 0.0000e+00 3.6600e+00 0.0000e+00 5.0000e-01 1.2500e+00 0.0000e+00 1.0000e+00
   1 8.4375e-01 3.3718e+00 0.0000e+00 5.9375e-01 5.4687e-01 0.0000e+00 6.2500e-02
   2 1.3345e+00 8.7152e-01 0.0000e+00 1.3354e+00 5.6415e-01 0.0000e+00 5.0000e-01
   3 7.3411e-01 7.2730e-01 0.0000e+00 2.0000e+00 8.0097e-01 0.0000e+00 1.0000e+00
   4 5.0097e-01 1.8183e-01 0.0000e+00 2.0000e+00 2.3429e+00 0.0000e+00 1.0000e+00
   5 5.0024e-01 4.5456e-02 0.0000e+00 2.0000e+00 4.6892e+00 0.0000e+00 1.0000e+00
   6 5.0006e-01 1.1364e-02 0.0000e+00 2.0000e+00 9.3801e+00 0.0000e+00 1.0000e+00
   7 5.0002e-01 2.8410e-03 0.0000e+00 2.0000e+00 1.8761e+01 0.0000e+00 1.0000e+00
   8 3.7501e-01 1.5981e-03 0.0000e+00 2.0000e+00 2.8142e+01 0.0000e+00 5.0000e-01
   9 3.0469e-01 8.9892e-04 0.0000e+00 2.0000e+00 3.9868e+01 0.0000e+00 5.0000e-01
   .
   .
   .
  52 1.0006e-01 9.0784e-11 0.0000e+00 2.0000e+00 1.4693e+05 0.0000e+00 2.5000e-01
  53 1.0005e-01 6.9506e-11 0.0000e+00 2.0000e+00 1.6792e+05 0.0000e+00 2.5000e-01
  54 1.0004e-01 5.3216e-11 0.0000e+00 2.0000e+00 1.9191e+05 0.0000e+00 2.5000e-01
  55 1.0003e-01 4.0743e-11 0.0000e+00 2.0000e+00 2.1933e+05 0.0000e+00 2.5000e-01
  56 1.0002e-01 3.1195e-11 0.0000e+00 2.0000e+00 2.5066e+05 0.0000e+00 2.5000e-01
  warning: matrix singular to machine precision, rcond = 8.33705e-17
-------------------------------------------------------------------------------

  Grdl(x,lm) =                          c(x) =

              -1.86265e-09                        +6.12843e-14
              +4.84932e-02                        +6.15064e-14
```

On a $c(x) = 0$ mais $\nabla_y l(x, \lambda) > 0$. Il n'y a donc pas de convergence de l'algorithme. En effet, les contraintes ne sont pas qualifiées puisqu'on n'a pas $\nabla e(x) = \sum_i \lambda_i \nabla c(x)$. En effet, les $\nabla c_i(x)$ sont dirigés suivant x alors que $\nabla e(x)$ est dirigé suivant y. C'est pourquoi on observe que $\|\lambda\| \to \infty$. Cela donne alors une matrice singulière à partir de la 56ème itération.

3.4.4 Cas test 3c



Convergence de l'algorithme: L'algorithme converge en 18 itérations:

iter	|gl|	|ce|	|x|	|lm|	alpha	phi	Q
0	2.2204e-16	3.8200e+00	3.0000e-01	7.5000e-01	1.0000e+00	7.6004e+00	3.8200e+00
1	2.2204e-16	2.0862e+00	1.3833e+00	7.5000e-01	1.2500e-01	4.0556e+00	1.4297e-01
2	0.0000e+00	2.1310e+00	1.7695e+00	7.5000e-01	5.0000e-01	2.2776e+00	4.8962e-01
3	0.0000e+00	1.3709e+00	2.0000e+00	7.5000e-01	1.0000e+00	1.8793e+00	3.0187e-01
4	0.0000e+00	3.4272e-01	2.0000e+00	7.5000e-01	1.0000e+00	1.1746e-01	1.8237e-01
5	0.0000e+00	8.5680e-02	2.0000e+00	7.5000e-01	1.0000e+00	7.3410e-03	7.2946e-01
6	0.0000e+00	2.1420e-02	2.0000e+00	7.5000e-01	1.0000e+00	4.5881e-04	2.9178e+00
7	0.0000e+00	5.3550e-03	2.0000e+00	7.5000e-01	1.0000e+00	2.8676e-05	1.1671e+01
8	0.0000e+00	1.3387e-03	2.0000e+00	7.5000e-01	1.0000e+00	1.7922e-06	4.6685e+01
9	0.0000e+00	3.3469e-04	2.0000e+00	7.5000e-01	1.0000e+00	1.1201e-07	1.8674e+02
10	0.0000e+00	8.3672e-05	2.0000e+00	7.5000e-01	1.0000e+00	7.0009e-09	7.4697e+02
11	0.0000e+00	2.0918e-05	2.0000e+00	7.5000e-01	1.0000e+00	4.3756e-10	2.9879e+03
12	0.0000e+00	5.2295e-06	2.0000e+00	7.5000e-01	1.0000e+00	2.7347e-11	1.1951e+04
13	0.0000e+00	1.3074e-06	2.0000e+00	7.5000e-01	1.0000e+00	1.7092e-12	4.7806e+04
14	0.0000e+00	3.2684e-07	2.0000e+00	7.5000e-01	1.0000e+00	1.0683e-13	1.9122e+05
15	0.0000e+00	8.1711e-08	2.0000e+00	7.5000e-01	1.0000e+00	6.6766e-15	7.6490e+05
16	0.0000e+00	2.0428e-08	2.0000e+00	7.5000e-01	1.0000e+00	4.1729e-16	3.0596e+06
17	0.0000e+00	5.1069e-09	2.0000e+00	7.5000e-01	1.0000e+00	2.6081e-17	1.2238e+07

Les contraintes ne sont pas qualifiées mais on a quand même l'existence de multiplicateurs optimaux. En effet, on a $\nabla e(x)$ dirigé suivant y et les $\nabla c_i(x)$ aussi.

Type de point stationnaire: (x, λ) est un point stationnaire et `eig(N'hlN) =`

Donc ...

4 Prise en compte de contraintes d'inégalité *(TP4)*

On souhaite désormais que notre optimiseur prenne en compte des contraintes d'inégalité. On résoudra alors dorénavant le problème d'optimisation

$$\begin{cases} \min e(x,y) = \sum_{i=1}^{n_b} L_i \dfrac{y_i + y_{i-1}}{2} \\ c_i(x,y) = (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 - L_i^2 = 0, \quad i = 1,...,n_b \\ c_{n_b+i+(j-1)n_n}(x,y) = r_i + x_j s_i - y_j \leqslant 0, \qquad\qquad i = 1,...,p, \quad j = 1,...,n_n \end{cases} \tag{P}$$

Les contraintes d'inégalités sont ici des fonctions affines. Elles représentent un plancher présent sous la chaîne. Les noeuds de la chaîne doivent alors se trouver au dessus de ce plancher affine par morceaux.

4.1 Méthode de Josephy-Newton

De manière analogue à ce qu'on faisait pour des contraintes d'égalité, nous cherchons des points stationnaires du problème.

Nous cherchons alors des points $z = (x, \lambda)$ vérifiant les conditions d'optimalité du premier ordre:

$$\begin{cases} \nabla_x l(x,\lambda) = 0 \\ c_E(x) = 0 \\ c_I(x) \leqslant 0, \lambda_I \in \mathbb{R}_+^{m_I}, \langle \lambda_I, c_I(x) \rangle = 0 \end{cases} \tag{KKT}$$

Dans la méthode de Newton, nous avions écrit la résolution des conditions d'optimalité comme le problème $F(z) = 0$. Nous ne pouvons pas écrire le problème actuel de cette manière.

Néanmoins, remarquons que $(x, \lambda_E, \lambda_I)$ est admissible $\Leftrightarrow z \in K = \mathbb{R}^n \times \mathbb{R}^{m_E} \times \mathbb{R}_+^{m_I}$ et que

z vérifie (KKT) $\Leftrightarrow -F(z) := -\begin{pmatrix} \nabla_x l(x,\lambda) \\ -c(x) \end{pmatrix} \in N_K(z)$.

On cherchera alors z tel que $0 \in F(z) + N_K(z)$.

Pour cela, cherchons une suite $\{z_k\}$ par récurrence telle que $0 \in F(z_k) + F'(z_k)(z_{k+1} - z_k) + N_K(z_k)$ et $\{z_k\}$ converge localement vers $z_\star$ solution de $0 \in F(z) + N_K(z)$.

Or on a: z_{k+1} solution de $0 \in F(z_k) + F'(z_k)(z_{k+1} - z_k) + N_K(z_k) \Leftrightarrow (x_{k+1}, \lambda_{k+1}) = (x_k + d, \lambda^{PQ})$ avec (d, λ^{PQ}) solution primale/duale du problème quadratique osculateur:

$$\text{(PQS)} \quad \begin{cases} \min\limits_{d \in \mathbb{R}^n} \nabla f(x_k)^\intercal d + \dfrac{1}{2} d^\intercal M_k d \\ c_E(x_k) + c_E'(x_k) d = 0 \\ c_I(x_k) + c_I'(x_k) d \leqslant 0 \end{cases}$$

Ainsi, dans chaque itération de l'algorithme de Newton, nous remplacerons la résolution d'un système linéaire par la résolution de (PQS).

4.2 Simulateur: ajout des variables d'inégalité

La signature de notre simulateur est désormais

```
[e,ce,ci,g,ae,ai,hl,indic] = chs(indic, xy, lme = [], lmi = [])
```

avec `e, ce, g, ae` déjà calculés dans 1.6.

4.2.1 Calcul de `ci`

Les contraintes d'inégalité s'écrivent de la forme $c_{n_b+i+(j-1)n_n}(x,y) = r_i + x_j s_i - y_j$

La variable globale `R` contient les pentes r_i et la variable globale `S` les ordonnées à l'origine s_i.
On calcule alors `ci` de la façon suivante:

$$c_I = \begin{pmatrix} r_1 \\ \vdots \\ r_1 \\ r_2 \\ \vdots \\ r_2 \\ \vdots \\ r_p \\ \vdots \\ r_p \end{pmatrix} + \begin{pmatrix} x_1 \times s_1 \\ \vdots \\ x_{n_n} \times s_1 \\ x_1 \times s_2 \\ \vdots \\ x_{n_n} \times s_2 \\ \vdots \\ x_1 \times s_p \\ \vdots \\ x_{n_n} \times s_p \end{pmatrix} - \begin{pmatrix} y_1 \\ \vdots \\ y_{n_n} \\ y_1 \\ \vdots \\ y_{n_n} \\ \vdots \\ y_1 \\ \vdots \\ y_{n_n} \end{pmatrix}$$

4.2.2 Calcul de `ai`

Notons $f_{i,j}(x,y) = C_{n_b+i+(j-1)n_n}(x,y) = r_j + x_i s_j - y_i \leqslant 0$
On a

$$\frac{\partial f_{i,j}(x,y)}{\partial x_k} = \begin{cases} s_k & \text{si } k = j \\ 0 & \text{sinon} \end{cases} \quad \text{et} \quad \frac{\partial f_{i,j}(x,y)}{\partial y_k} = \begin{cases} -1 & \text{si } k = j \\ 0 & \text{sinon} \end{cases}$$

Ce qui donne par exemple pour $n_n = 4$ et $p = 2$:

$$\left(\begin{array}{cccc|cccc} s_1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & s_1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & s_1 & 0 & 0 & 0 & -1 \\ s_2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & s_2 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & s_2 & 0 & 0 & 0 & -1 \end{array}\right)$$

4.2.3 Calcul de `hl`

On a $l(x, \lambda_E, \lambda_I) = f(x) + \lambda_E^\intercal c_E(x) + \lambda_I^\intercal c_I(x)$
Donc $\nabla_x l(x, \lambda_E, \lambda_I) = \nabla_x f(x) + \lambda_E c'_E(x) + \lambda_I c'_I(x)$
Et alors $\nabla_x x^2 l(x, \lambda_E, \lambda_I) = \nabla_x x^2 f(x) + \lambda_E \nabla_x x^2 c_E(x) + \lambda_I \nabla_x x^2 c_I(x)$
Mais comme les contraintes C_I sont affines en x alors $\nabla_x x^2 c_I(x) = 0$ et donc $\boxed{\nabla_x x^2 l(x, \lambda_E, \lambda_I) = \nabla_x x^2 l(x, \lambda_E)}$.
`hl` reste donc **inchangé** avec l'ajout des contraintes d'inégalité.

4.3 Implémentation de l'optimiseur

4.3.1 Résolution de (PQS)

La fonction `qp` d'octave permet de résoudre des problèmes quadratiques de la forme

$$\begin{cases} \texttt{min 0.5 x'*H*x + x'*q} \\ \texttt{A*x = b} \\ \texttt{lb <= x <= ub} \\ \texttt{Alb <= Ain*x <= Aub} \end{cases}$$

Ici il s'agira de résoudre (PQS):

- $H = M_k =$ `M` (calculé avec cholmod)
- $x = d_k =$ `dir`
- `q` $= \nabla_x f(x_k) =$ `g`
- $A = c'_E =$ `ae`
- $b = -c_E =$ `-ce`
- `Ain` $= c'_I =$ `ai`
- `Aub` $= -c_I =$ `-ci`
- $lb = -\infty =$ `-Inf`
- $ub = +\infty =$ `-Inf`
- `Alb` $= -\infty =$ `-Inf`

4.3.2 Approximation définie positive du hessien du lagrangien *(Q4.1)*

Pour s'assurer que le (PQO) soit borné, il faudrait que `hl` soit semi-défini positif. Or ceci n'est pas garanti, nous l'approchons donc par une matrice définie positive M. Ceci ne change pas les solutions trouvées. Nous pouvons donc être tenté de choisir I comme approximation définie positive de `hl`. Le problème à résoudre devient alors

$$\begin{cases} \min & \nabla f(x_k)^\intercal d_k + 0.5\|d_k\|_2 \\ & c_E(x_k) + c'_E(x_k)d_k = 0 \\ & c_I(x_k) + c'_I(x_k)d_k \leqslant 0 \end{cases}$$

L'algorithme devrait quand même renvoyer les bonnes solutions. Néanmoins la convergence devient beaucoup plus lente de cette façon. En effet, plus l'approximation de `hl` est bonne et plus l'algorithme convergera rapidement. Nous avons donc intérêt de bien la choisir.

4.3.3 Factorisation de Cholesky

Une façon d'approcher `hl` par une matrice définie positive est de faire une factorisation de cholesky. La fonction `cholmod` fait une factorisation de cholesky.

`[L, d, flag] = cholmod (M, small, big)`
Renoie L et d tq `M + E = L*D*L'`

$$flag = \begin{cases} 0 \text{ c good} \\ 1 \text{ M est pas carrée ou vide} \\ 2 \text{ small ou big négative ou } \sqrt{small} > big \end{cases}$$

On prendre : $small = 1.e-5$ et $big = 1.e+5$

4.3.4 Calcul du premier multiplicateur *(Q4.2)*

Soit x un point stationnaire de notre problème, alors il existe $\lambda = (\lambda_E, \lambda_I)$ tel que

$$\begin{cases} \nabla f(x) + c'(x)^\intercal \lambda = 0 \\ c_E(x) = 0 \\ \sum_{i=n_b}^{pn_n} \lambda_i c_i(x) = 0, \lambda_I \geqslant 0, c_I(x) \leqslant 0 \end{cases}$$

On peut alors résoudre le problème d'optimisation

$$\begin{cases} \min\limits_{\lambda \in \mathbb{R}_+^{m_E+m_I}} & \|\nabla f(x) + c'(x)^\intercal \lambda\|_2^2 \\ & c_E(x) = 0 \\ & \sum\limits_{i=m_E+1}^{m_E+m_I} \lambda_i c_i(x) = 0 \\ & \lambda_I \geqslant 0, \quad c_I(x) \leqslant 0 \end{cases}$$

Pour ceci, on utilise la fonction `qp`.
On a

$$\|\nabla f(x) + c'(x)^\intercal \lambda\|_2^2 = \|\nabla f(x)\| + 2\left(\lambda^\intercal \times c'(x)\nabla f(x) + 0.5\lambda^\intercal c'(x)c'(x)^\intercal \lambda\right)$$

On mettra alors en entrée

$$\begin{cases} \texttt{x} = \lambda = \texttt{lm} \\ \texttt{H} = c'(x)c'(x)^\intercal = \texttt{[ae;ai]*[ae;ai]'} \\ \texttt{q} = c'(x)\nabla f(x) = \texttt{[ae;ai]*g} \end{cases} \quad \begin{cases} \texttt{A} = \begin{pmatrix} 0 & c_I(x)^\intercal \\ 0 & 0 \end{pmatrix} = \texttt{[zeros(1,me) ci';zeros(me+mi,mi)]} \\ \texttt{b} = \begin{pmatrix} 0 \\ c_E(x) \end{pmatrix} = \texttt{[0;ce]} \end{cases}$$

$\texttt{lb} = 0$
$\texttt{up} = +\infty = \texttt{Inf}$
$\texttt{A\_lb} = c_I(x) = \texttt{ci}$
$\texttt{A\_in} = 0 = \texttt{0}$
$\texttt{A\_ub} = +\infty = \texttt{Inf}$

4.3.5 Convergence quadratique ?? *(Q4.5)*

4.4 Etude de plusieurs conditions initiales

4.4.1 Cas test 4.a *(Q4.4)*

Dans un premier temps, nous testons un cas sans contraintes d'inégalité afin d'observer la différence entre cet algorithme qui résoud un problème quadratique à chaque itération et l'algorithme de Newton initialement implémenté en *(2.1)*. Nous reprenons donc le cas test 2.2.2. L'algorithme de Newton trouvait alors un maximum local.

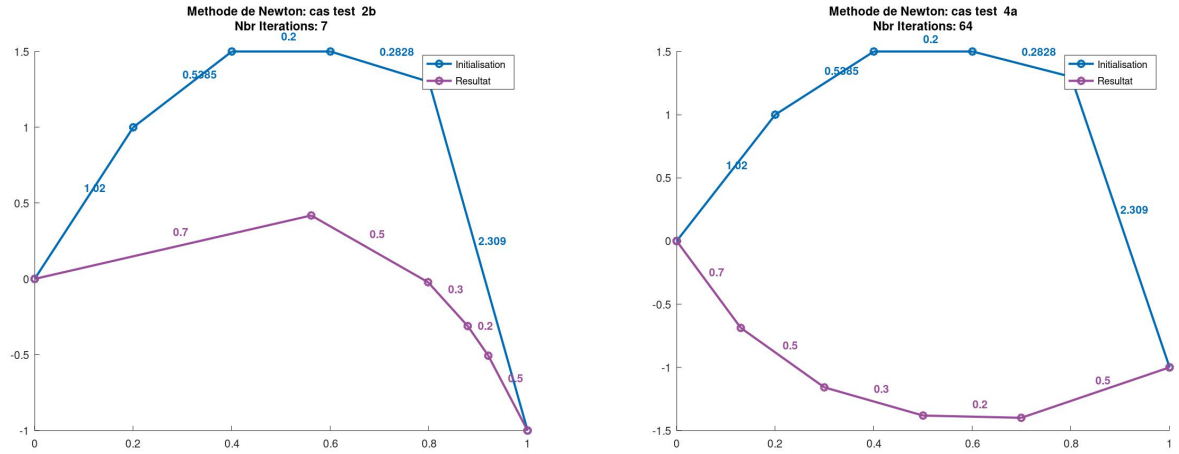


Figure 1: Méthode de Newton (à gauche) et de Josephy-Newton (à droite)

On trouve cette fois ci un minimum global. Pourquoi *(Q4.4)* ? Comparons les deux algorithmes :

Nous souhaitons résoudre le problème $\begin{cases} \min & f(x) \\ & c_E(x) = 0 \end{cases}$

1. itéré (x_k, λ_k)
2. on cherche une direction de descente (d_k, λ_k^{PQ}) telle que

Algorithme de Newton

$$\begin{pmatrix} \nabla^2_{xx} l(x_k, \lambda_k) & c'_E(x) \\ c'_E(x)^\intercal & 0 \end{pmatrix} \begin{pmatrix} d_k \\ \lambda_k^{PQ} \end{pmatrix} = - \begin{pmatrix} \nabla f(x) \\ c_E(x) \end{pmatrix}$$

C'est-à-dire

$$\begin{cases} \nabla^2_{xx} l(x_k, \lambda_k) d_k + c'_E(x_k) \lambda_k^{PQ} = -\nabla f(x_k) \\ c'_E(x_k)^\intercal d_k = -c_E(x_k) \end{cases}$$

Algorithme de Josephy-Newton

$$\begin{cases} \min & \nabla f(x_k)^\intercal d_k + 0.5 d_k^\intercal M_k d_k \\ & c_E(x_k) + c'_E(x_k) d_k = 0 \end{cases} \quad \text{(PQO)}$$

On résoud donc les conditions d'optimalité:

$$\begin{cases} \nabla f(x_k) + \nabla^2_{xx} l(x_k, \lambda_k) d_k + c'_E(x_k) \lambda_k^{PQ} = 0 \\ c_E(x_k) + c'_E(x_k)^\intercal d_k = 0 \end{cases}$$

On résoud donc le même système *(conditions d'optimalité du PQO)* dans les deux cas. La différence se trouve dans le fait que dans l'algorithme de Josephy-Newton, on cherche une direction de descente qui minimise l'approximation quadratique de f (+ un terme de pénalisation). En effet, $M_k \simeq \nabla^2_{xx} l(x_k, \lambda_k)$ et

$$\nabla f(x_k)^\intercal d_k + 0.5 d_k^\intercal M_k d_k \simeq \underbrace{f(x_k)}_{=0} + \nabla f(x_k)^\intercal d_k + 0.5 d_k^\intercal \nabla^2 f(x_k) d_k + 0.5 d_k^\intercal \sum_i \lambda_{ki} \nabla^2 c_i(x_k) d_k$$

Conditions initiales:

```
x = (0.2, 0.4, 0.6, 0.8)'
y = (1.0, 1.5, 1.5, 1.3)'
```

Solution:

```
x = ( 0.13013   0.29951   0.49992   0.69912)'
y = (-0.68780 -1.15824 -1.38149 -1.39934)'
```

Convergence de l'algorithme: On observe la convergence de l'algorithme en 63 itérations.

iter	|gl|	|ce|	|ci|	|x|	|lme|	|lmi|
0	1.0947e-01	5.0800e+00	0.0000e+00	1.5000e+00	4.9918e-01	0.0000e+00
1	9.0060e+00	2.2968e+00	0.0000e+00	1.3468e+00	3.2649e+00	0.0000e+00
2	4.0232e+00	1.5539e+00	0.0000e+00	1.2738e+00	6.0979e+00	0.0000e+00
3	7.5486e+00	6.1972e-01	0.0000e+00	1.1178e+00	8.9987e+00	0.0000e+00
4	3.7764e+00	3.1464e-01	0.0000e+00	9.5537e-01	5.4832e+00	0.0000e+00
5	1.2454e+00	2.8272e-01	0.0000e+00	9.4263e-01	2.5770e+00	0.0000e+00
.						
.						
.						
58	1.2082e+00	3.3103e-04	0.0000e+00	1.3989e+00	9.4859e-01	0.0000e+00
59	1.2050e+00	1.6556e-04	0.0000e+00	1.3989e+00	9.3878e-01	0.0000e+00
60	1.2030e+00	8.2801e-05	0.0000e+00	1.3990e+00	9.3280e-01	0.0000e+00
61	1.2018e+00	4.1411e-05	0.0000e+00	1.3991e+00	9.2931e-01	0.0000e+00
62	1.2012e+00	2.0714e-05	0.0000e+00	1.3992e+00	9.2734e-01	0.0000e+00
63	1.2008e+00	1.0363e-05	0.0000e+00	1.3992e+00	9.2627e-01	0.0000e+00

On a $\|\nabla^2_{xx} l(x,\lambda)\| > 0$.

4.4.2 Cas test 4.b

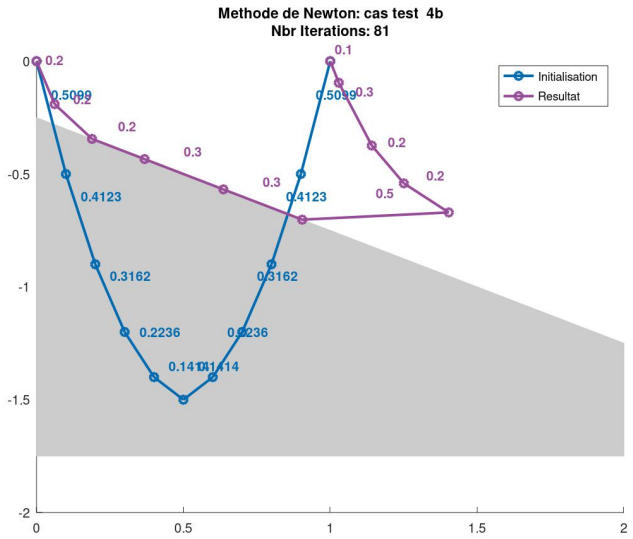
Étudions maintenant un premier cas avec plancher.

Conditions initiales:

```
x = ( 0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9)'
y = (-0.5 -0.9 -1.2 -1.4 -1.5 -1.4 -1.2 -0.9 -0.5)'
```

Solution:

```
x = ( 0.06  0.19  0.37  0.64  0.90  1.40  1.25  1.14  1.03)'
y = (-0.19 -0.34 -0.43 -0.57 -0.70 -0.67 -0.54 -0.37 -0.10)'
```



Convergence de l'algorithme: On observe la convergence de l'algorithme en 80 itérations.

iter	|gl|	|ce|	|ci|	|x|	|lme|	|lmi|
0	4.0000e-01	2.5000e-01	1.0000e+00	1.5000e+00	0.0000e+00	0.0000e+00
1	2.1342e+04	4.3620e+07	2.0094e+03	5.4555e+03	1.1432e+00	1.0468e-01
2	2.3714e+06	1.0905e+07	1.6133e+03	2.7354e+03	1.1685e+04	7.0694e+03
3	1.9741e+05	2.7268e+06	9.9602e+02	1.3783e+03	2.0710e+04	2.4852e+04

```
 4 1.6501e+04 6.8268e+05 6.5309e+02 6.8672e+02 1.3320e+04 0.0000e+00
 5 9.8231e+03 1.7072e+05 4.7999e+02 3.8136e+02 1.7255e+04 3.9861e+01
 .
 .
 .
75 7.2003e-01 4.7009e-05 6.6736e-01 1.3978e+00 5.4438e+00 5.2226e-01
76 7.1722e-01 3.6073e-05 6.6762e-01 1.3993e+00 5.4003e+00 5.2034e-01
77 7.1426e-01 2.5863e-05 6.6789e-01 1.4005e+00 5.3560e+00 5.1824e-01
78 7.1087e-01 1.9663e-05 6.6807e-01 1.4015e+00 5.3179e+00 5.1334e-01
79 7.0899e-01 1.3963e-05 6.6836e-01 1.4022e+00 5.2933e+00 5.0995e-01
80 7.0734e-01 1.0448e-05 6.6861e-01 1.4029e+00 5.2741e+00 5.0645e-01
```

4.4.3 Cas test 4.c

Conditions initiales:

```
x = ( 0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9)'
y = (-0.5 -0.9 -1.2 -1.4 -1.5 -1.4 -1.2 -0.9 -0.5)'
```

L'algorithme s'arrête avant même la première itération avec `info.status = 6`. Ce qui signifie que le premier (PQS) n'a pas de solution.

Le premier (PQS) à résoudre est $\begin{cases} \min\limits_{d\in\mathbb{R}^n} \nabla f(x_0)^\intercal d + \dfrac{1}{2}\|d\| \\ c_E(x_0) + c'_E(x_0)d = 0 \\ c_I(x_0) + c'_I(x_0)d \leqslant 0 \end{cases}$

Avec

- $\nabla f(x_0) = \begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\0\\0\\0.2\\0.2\\0.25\\0.3\\0.4\\0.35\\0.2\\0.25\\0.2 \end{pmatrix}$
- $c_E(x_0) = \begin{pmatrix} 0.22\\0.13\\0.06\\0.04\\0.07\\0.23\\0.01\\0.06\\0.08\\0.25 \end{pmatrix}$
- $c_I(x_0) = \begin{pmatrix} 0.2\\0.4\\0.8\\0.9\\1\\0.9\\0.6\\0.4\\-0.2\\0\\0.55\\0.7\\0.95\\1\\0.85\\0.7\\0.25\\0 \end{pmatrix}$
- $c'_I(x_0) = \begin{pmatrix} -0.5I_n & -I_n \\ 0 & -I_n \end{pmatrix}$
- $c'_E(x_0) =$

$$\begin{pmatrix} 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8 & -0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.6 & -0.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4 & -0.4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & -0.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.2 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.4 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.6 & 0.6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.8 & 0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

On a alors notamment $\begin{cases} 0.22 + 0.2d_1 + d_{10} = 0 \\ 0.13 - 0.2d_1 + 0.2d_2 + 0.8d_{10} - 0.8d_{11} = 0 \\ 0.2 - 0.5d_1 - d_{10} \leqslant 0 \\ -d_{10} \leqslant 0 \end{cases}$

Donc on a $\begin{cases} d_{10} = -0.22 - 0.2d_1 \\ 0.13 - 0.2d_1 + 0.2d_2 + 0.8d_{10} - 0.8d_{11} = 0 \\ d_1 \geqslant 0.4 - 2d_{10} \\ d_{10} \geqslant 0 \end{cases}$

Or on a $0 \leqslant d_{10} = -0.22 - 0.2d_1$ donc $\boxed{d_1 \leqslant -1.1}$

De plus, $0.4 - 2d_{10} \leqslant 0.4$ donc il existe $\delta \leqslant 0.4$ tel que $d_1 \geqslant \delta$

5 Version quasi-newtonienne *(TP4)*

Formule de BFGS :

$$M_{k+1} = M_k - \frac{M_k \delta_k \delta_k^\intercal M_k}{\delta_k^\intercal M_k \delta_k} + \frac{\gamma_k \gamma_k^\intercal}{\gamma_k^\intercal \delta_k}$$

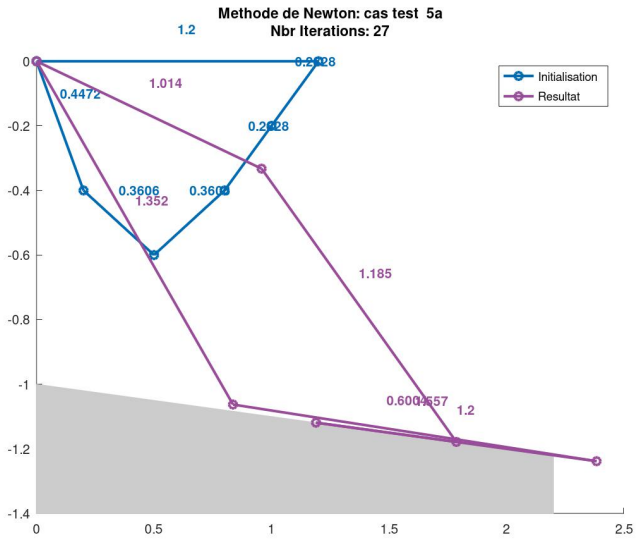
La variation de x : $\delta = x_{k+1} - x_k$
La variation du gradient du lagrangien : $\gamma_k^\ell = \nabla \ell(x_{k+1}, \lambda_{k+1}) - \nabla \ell(x_k, \lambda_{k+1})$
On veut avoir $\gamma_k^\intercal \delta_k > 0$
Correction de Powel : $\gamma_k = (1-\theta) M_k \delta_k + \theta \gamma_k^\ell$
avec $\theta \in]0, 1]$ maximale tel que : $\gamma^\intercal \delta_k \geqslant 0.2 \delta_k^\intercal M_k \delta_k$

$$\theta = \begin{cases} 0.8 \dfrac{\delta_k^\intercal M_k \delta_k}{\delta_k^\intercal M_k \delta_k - (\gamma_k^\ell)^\intercal \delta_k} & \text{si } (\gamma_k^\ell)^\intercal \delta_k < 0.2 \delta_k^\intercal M_k \delta_k \\ 1 & \text{sinon} \end{cases}$$

Pour la première iteration on prend : $M_1 = I$ pour calculer x_2, puis $M_1 = \underbrace{\dfrac{\|\gamma_1\|_2^2}{\gamma_1^\intercal \delta_1}}_{=\eta_1} I$ pour le calcule de M_2

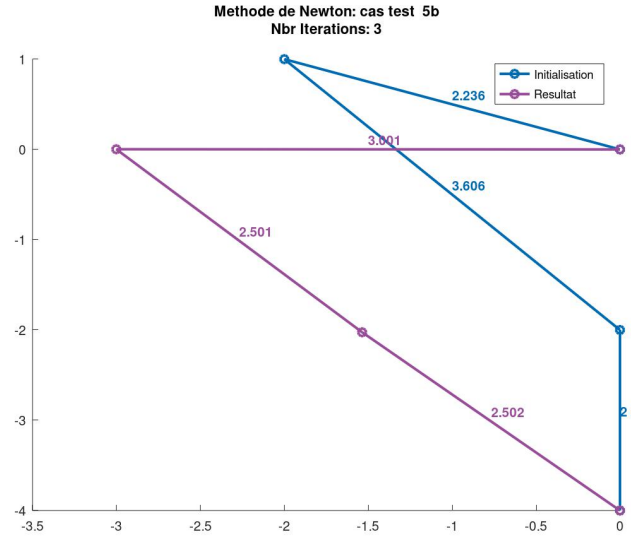
5.1 Etude de plusieurs conditions initiales

5.1.1 Cas test 5.a



A la 27ème itération, le PQS ne converge pas avant 200 itérations, on sort donc de l'algo.

5.1.2 Cas test 5.b



A la 3ème itération, le PQS n'est pas réalisable et on sort donc de l'algorithme.

5.1.3 Cas test 5.c