Projet d'OPT-202 : Équilibre d'une chaîne articulée sur un plancher

Leonardo Martins Bianco - Guillaume Lambert

25 février 2021

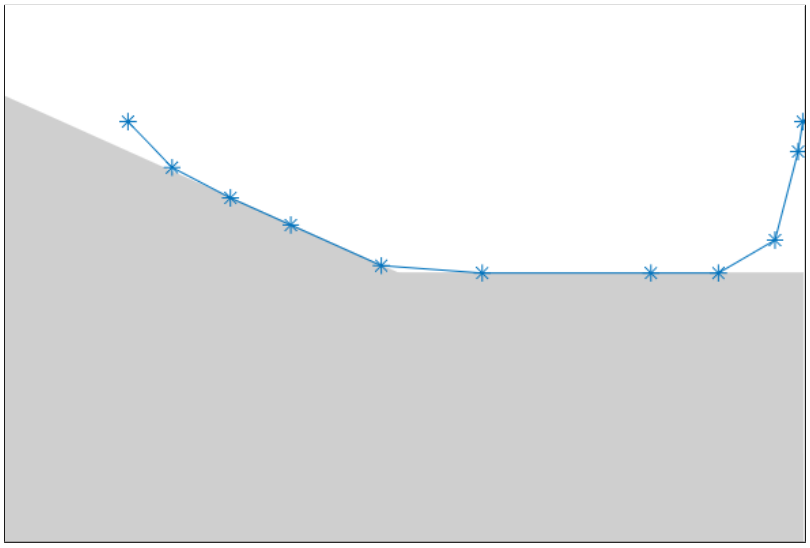


Table des matières

Introduction

Ce projet d'optimisation considère le problème de trouver la position d'équilibre statique d'une chaîne formée de barres rigides, contenue dans un plan vertical, fixée à ses deux bouts et sous contraintes d'inégalités qui représentent un plancher en-dessous de la chaîne. Le langage de programmation utilisé est *Matlab*.

Dans un premier temps, on modélise le nouveau problème comprenant les contraintes d'inégalités via le *problème quadratique osculateur* et on met à jour notre ancien optimiseur pour résoudre à chaque itération ce nouveau système. Ensuite, on discute plus en détails le choix de la matrice qui remplace la Hessienne dans le PQO, un critère pour décider si la solution trouvé est un minimum local, et des aspects de convergence (vitesse et solutions trouvées par rapport aux algorithmes précédents).

Dans un second temps, on introduit une version quasi-newtonienne afin de calculer la matrice approchant le hessien du lagrangien. Cela nous mènera à la formule de BFGS. Ensuite, on étudiera le caractère bien posé du programme quadratique osculateur et aussi la formule de BFGS, notamment son intérêt, sa construction et son initialisation.

À la fin de chaque partie, nous étudions des cas-tests pour vérifier notre code et les propriétés théoriques discutées.

1 Prise en compte des contraintes d'inégalités

1.1 Introduction du nouveau problème

On souhaite introduire des contraintes d'inégalités au problème avec contraintes d'égalités :

$$(P) \quad \begin{cases} \min e(\mathrm{x},\mathrm{y}) \\ c_i(\mathrm{x},\mathrm{y}) = 0, \quad i = 1,\dots,N_b \end{cases}$$

On va noter c_E la fonction associée aux contraintes d'égalités et c_I celle associée aux contraintes d'inégalités. L'ajout de contraintes d'inégalités dans notre problème s'interprète physiquement par l'ajout d'un plancher sous la chaîne articulée.

Le plancher est convexe linéaire par morceaux et s'exprime comme l'enveloppe supérieur d'un nombre fini p de fonctions affines. L'intérêt de l'ajout d'un plancher est que la chaîne considérée ne pourra pas prendre la position d'équilibre associée au problème sans le plancher. On considère la plancher infiniment glissant : on ne prend donc pas en compte d'éventuels frottements entre la chaîne et le plancher.

Le plancher est défini dans le plan (x,y) par la fonction affine φ définie par

$$\begin{aligned} \varphi : \mathbb{R} &\to \mathbb{R}^p \\ \mathrm{x} &\mapsto r + \mathrm{x}s \end{aligned}$$

où $r \in \mathbb{R}^p$ est le vecteur des ordonnées à l'origine des fonctions affines définissant le plancher et $s \in \mathbb{R}^p$ est le vecteur des pentes associées.

On note C le convexe des points dans $\mathbb{R}^2$ qui vérifient les contraintes d'inégalités c_I, c'est-à-dire qui sont au-dessus du plancher. Avec ces notations, on a donc

$$C = \{(\mathrm{x}, \mathrm{y}) \in \mathbb{R} \times \mathbb{R} | y\overline{e} \geq \varphi(\mathrm{x})\}$$

avec $\overline{e} := (1; 1; ..., 1) \in \mathbb{R}^p$.

Pour que le problème soit bien posé, les points de fixation $(0, 0)$ et (a, b) doivent être dans C, c'est-à-dire

$$r \leq 0 \text{ et } r + as \leq be$$

Etant donné que la chaîne est affine par morceaux et que le plancher est convexe, si tous les noeuds sont dans C, alors la chaîne sera entièrement contenue dans C. Autrement dit

$$\forall i \in \{1, ..., n_n\}, \forall j \in \{1, ..., p\},\ c_{n_b+(j-1)n_n+i}(\mathrm{x}, \mathrm{y}) \equiv r_j + \mathrm{x}_i s_j - \mathrm{y}_i \leq 0$$

Le problème à résoudre est donc de la forme

$$\begin{cases} \min e(\mathrm{x}, \mathrm{y}) \\ c_i(\mathrm{x}, \mathrm{y}) = 0, \quad i = 1, \ldots, N_b \\ c_i(\mathrm{x}, \mathrm{y}) \leq 0, \quad i \in [n_b + 1, \ldots n_b + pn_n] \end{cases} \tag{P$_2$}$$

où les contraintes d'égalité c_E sont les n_b premières contraintes et les contraintes d'inégalités c_I sont les suivantes.

1.2 Mise à jour du simulateur

Les modifications du code de `chs` correspondent à l'ajout des contraintes d'inégalités. En entrée, on considère maintenant des multiplicateurs de Lagrange d'égalités et aussi d'inégalités (respectivement `lme` et `lmi`). En sortie, on renvoie en plus

— `ci` : la valeur contraintes d'inégalités $c_I(x)$
— `ai` : la jacobienne des contraintes d'inégalités $c_I'(x)$

On modifie aussi le calcul du hessien du lagrangien étant donné que le lagrangien est modifié.

Simulateur chs Voici la nouvelle documentation de la fonction simulateur `chs` :

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

Paramètres d'entrée

— `indic` : entier pilotant le comportement du simulateur
  — 1 : tracé de la chaîne
  — 2 : calcule `e`, `ce` et `ci`
  — 4 : calcule `e`, `ce`,`ci`, `g`, `ae` et `ai`
  — 5 : calcule `hl`

— 6 : affiche l'étude de l'exactitude du gradient de `e`

— `xy` : vecteur-colonne des abscisses puis des ordonnées des points d'articulation de la chaîne, $\mathtt{xy} = (\mathtt{x}_1, ..., \mathtt{x}_{n_n}, \mathtt{y}_1, ..., \mathtt{y}_{n_n})$. Il est de dimension $(2n_n, 1)$.

— lme : vecteur-colonne des multiplicateurs de Lagrange pour les contraintes d'égalité $\lambda_E = \{\lambda_i\}_{i=1}^{n_b}$. Il est de dimension $(n_b, 1)$.

— lmi : vecteur-colonne des multiplicateurs de Lagrange pour les contraintes d'inégalités $\lambda_I = \{\lambda_i\}_{i=n_b+1}^{n_b+n_n}$. Il est de dimension $(n_n, 1)$.

Valeurs renvoyées

Nous présentons les valeurs renvoyées :

Calcul des contraintes d'inégalités : `ci` est un vecteur-colonne de dimension pn_n contenant les contraintes d'inégalités c_I. On les calcule par

$$\forall i \in \{1, ..., n_n\}, \forall j \in \{1, ..., p\},\ c_{n_b+(j-1)n_n+i}(\mathtt{x}, \mathtt{y}) \equiv r_j + \mathtt{x}_i s_j - \mathtt{y}_i \leq 0$$

C'est la fonction `fun_ci` qui calcule `ci`.

Calcul des contraintes d'inégalités : `ai` est une matrice de pn_n lignes et $2n_n$ colonnes qui contient la jacobienne des contraintes d'inégalités en `xy`. On la calcule comme ceci : soient $i \in \{1, ..., n_n\}$, $j \in \{1, ..., p\}$

$$
c'_I(\mathrm{x},\mathrm{y}) = \begin{pmatrix}
c'_{n_b+(1-1)n_n+1}(\mathrm{x},\mathrm{y}) \\
c'_{n_b+(1-1)n_n+2}(\mathrm{x},\mathrm{y}) \\
\vdots \\
c'_{n_b+(1-1)n_n+n_n}(\mathrm{x},\mathrm{y}) \\
c'_{n_b+(2-1)n_n+1}(\mathrm{x},\mathrm{y}) \\
\vdots \\
c'_{n_b+(2-1)n_n+n_n}(\mathrm{x},\mathrm{y}) \\
\vdots \\
\vdots \\
\vdots \\
c'_{n_b+(p-1)n_n+1}(\mathrm{x},\mathrm{y}) \\
\vdots \\
c'_{n_b+(p-1)n_n+n_n}(\mathrm{x},\mathrm{y})
\end{pmatrix}
$$

$$
= \begin{pmatrix}
\nabla r_1 + x_1 s_1 - y_1 \\
\nabla r_1 + x_2 s_1 - y_2 \\
\vdots \\
\nabla r_1 + x_{n_n} s_1 - y_{n_n} \\
\nabla r_2 + x_1 s_2 - y_1 \\
\vdots \\
\nabla r_2 + x_{n_n} s_2 - y_{n_n} \\
\vdots \\
\vdots \\
\vdots \\
\nabla r_p + x_1 s_p - y_1 \\
\vdots \\
\nabla r_p + x_{n_n} s_p - y_{n_n}
\end{pmatrix}
$$

$$
= \begin{pmatrix}
s_1 & 0 & \dots & \dots & 0 & -1 & 0 & \dots & \dots & 0 \\
0 & s_1 & 0 & \dots & 0 & 0 & -1 & 0 & \dots & 0 \\
0 & \ddots & \ddots & \ddots & \vdots & 0 & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & \ddots & s_1 & 0 & \vdots & \ddots & \ddots & -1 & 0 \\
0 & \dots & \dots & 0 & s_1 & 0 & \dots & \dots & 0 & -1 \\
s_2 & 0 & \dots & \dots & 0 & -1 & 0 & \dots & \dots & 0 \\
0 & s_2 & 0 & \dots & 0 & 0 & -1 & 0 & \dots & 0 \\
0 & \ddots & \ddots & \ddots & \vdots & 0 & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & \ddots & s_2 & 0 & \vdots & \ddots & \ddots & -1 & 0 \\
0 & \dots & \dots & 0 & s_2 & 0 & \dots & \dots & 0 & -1 \\
\end{pmatrix}
$$

$$
\begin{pmatrix}
s_p & 0 & \dots & \dots & 0 & -1 & 0 & \dots & \dots & 0 \\
0 & s_p & 0 & \dots & 0 & 0 & -1 & 0 & \dots & 0 \\
0 & \ddots & \ddots & \ddots & \vdots & 0 & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & \ddots & s_p & 0 & \vdots & \ddots & \ddots & -1 & 0 \\
0 & \dots & \dots & 0 & s_p & 0 & \dots & \dots & 0 & -1
\end{pmatrix}
$$

C'est la fonction `fun_ai` qui calcule `ai`.

Calcul du hessien : Le hessien du lagrangien est modifié pour prendre en compte les contraintes d'inégalités :

$$\nabla^2_{xx}\ell(x,\lambda) = \nabla^2 e(x) + \sum_{i\in E}\lambda_i\nabla^2 c_i(x) + \sum_{i\in I}\lambda_i\nabla^2 c_i(x)$$

Cependant, la matrice hessienne $\nabla^2 c_I(x)$ est la matrice nulle étant donné que les contraintes d'inégalités sont linéaires. Le hessien du lagrangien n'est donc pas modifié.

1.3 Problème quadratique osculateur

On va améliorer la méthode de résolution locale basée sur l'algorithme de Newton pour prendre en compte les contraintes d'inégalités. On introduit donc l'algorithme d'*optimisation quadratique successive* (OQS). On le qualifie d'algorithme *successif* car on résout, à chaque itération, un problème d'optimisation quadratique. Cet algorithme est lui-aussi un algorithme primal-dual qui génère une suite $\{(x_k,\lambda_k)\}_k \subset \mathbb{R}^n\times\mathbb{R}^m$. A l'itération k, on construit le couple (x_{k+1},λ_{k+1}) par :

$$x_{k+1} = x_k + d_k \text{ et } \lambda_{k+1} = \lambda_k^{PQ}$$

où (d_k,λ_k^{PQ}) est une solution primale-duale du *problème quadratique osculateur* :

$$\begin{cases} \min_{d\in\mathbb{R}^n} \nabla f(x_k)^\top d + \frac{1}{2}d^\top M_k d \\ c_E(x_k) + c'_E(x_k)d = 0 \\ c_I(x_k) + c'_I(x_k)d \le 0 \end{cases}$$

où M_k est le hessien du lagrangien $\nabla^2_{xx}\ell(x_k,\lambda_k)$ ou une approximation de celui-ci.

On se demande maintenant comment et pourquoi on se ramène à l'utilisation de cet algorithme.

La condition d'optimalité d'ordre 1 de KKT pour le problème $(\mathcal{P}_{EI})$ est la suivante : s'il existe $x_\star\in\mathbb{R}^n$ solution de $(\mathcal{P}_{EI})$, alors il existe un multiplicateur de Lagrange optimal $\lambda_\star\in\mathbb{R}^m$ tel que

$$\begin{cases} \nabla f(x_\star) + c'(x_\star)^\top\lambda_\star = 0 \\ c_E(x_\star) = 0 \\ 0 \le (\lambda_\star)_I \perp c_I(x_\star) \le 0 \end{cases}$$

On peut réécrire cela sous la forme suivante :

$$K \ni z_\star \perp F(z_\star) \in K^+$$

où $F : \mathbb{R}^n\times\mathbb{R}^m \to \mathbb{R}^n\times\mathbb{R}^m$ est définie en $z=(x,\lambda)$ par

$$F(z) = \begin{pmatrix} \nabla_x\ell(x,\lambda) \\ -c(x) \end{pmatrix}$$

et $K = \mathbb{R}^n\times(\mathbb{R}^{m_E}\times\mathbb{R}^{m_I}_+)$ (et donc $K^+ = \{0_{\mathbb{R}^n}\}\times(\{0_{\mathbb{R}^{m_E}}\}\times\mathbb{R}^{m_I}_+)$).

En effet,

— $z_\star \in K$ correspond aux domaines de définition de $x_\star$ et $\lambda_\star$ (en particulier $(\lambda_\star)_I \geq 0$).
— $F(z_\star) \in K^+$ correspond à
  — $\nabla_x \ell(x, \lambda) = \nabla f(x_\star) + c'(x_\star)^\top \lambda_\star = 0$
  — $c_E(x_\star) = 0$
  — $c_I(x_\star) \leq 0$
— $z_\star \perp F(z_\star)$ correspond à

$$x_\star^\top \underbrace{\nabla_x \ell(x, \lambda)}_{=0} - \lambda_E^\top \underbrace{c_E(x_\star)}_{=0} - (\lambda_\star)_I^\top c_I(x_\star) = 0 \iff (\lambda_\star)_I^\top c_I(x_\star) = 0 \iff (\lambda_\star)_I^\top \perp c_I(x_\star)$$

La prochaine étape consiste à linéariser F au point z_{k+1} dans l'expression $K \ni z_\star \perp F(z_\star) \in K^+$:

$$K \ni z_{k+1} \perp F(z_k) + F'(z_k)(z_{k+1} - z_k) \in K^+$$

Si on note $(d_k, \mu_k) = z_{k+1} - z_k$, on obtient

$$K \ni z_{k+1} \perp F(z_k) + F'(z_k)(d_k, \mu_k) \in K^+$$

$$\iff K \ni z_{k+1} \perp \begin{pmatrix} \nabla_x f(x_k) + c'(x_k)^\top \lambda_k + H_k(x_{k+1} - x_k) + c'(x_k)^\top(\lambda_{k+1} - \lambda_k) \\ -c(x_k) - c'(x_k) d_k \end{pmatrix} \in K^+$$

On pose $H_k = \nabla_{xx} \ell(x_k, \lambda_k)$ et on simplifie les termes en λ_k :

$$K \ni z \perp \begin{pmatrix} \nabla_x f(x_k) + H_k(x - x_k) + c'(x_k)^\top \lambda_{k+1} \\ -c(x_k) - c'(x_k) d_k \end{pmatrix} \in K^+$$

Cela revient à

$$\begin{cases} \nabla_x f(x_k) + H_k d_k + c'(x_k)^\top \lambda_{k+1} = 0 \\ c_E(x_k) + c'_E(x_k) d_k = 0 \\ 0 \leq (\lambda_{k+1})_I \perp c_I(x_k) + c'_I(x_k) d_k \leq 0 \end{cases}$$

Ce système en (d_k, λ_{k+1}) n'est pas facile à résoudre mais on remarque qu'il a la forme de la condition d'optimalité de KKT d'un certain problème. Ce problème est le *problème quadratique osculateur* suivant

$$\begin{cases} \min_{d \in \mathbb{R}^n} \nabla f(x_k)^\top d + \frac{1}{2} d^\top M_k d \\ c_E(x_k) + c'_E(x_k) d = 0 \\ c_I(x_k) + c'_I(x_k) d \leq 0 \end{cases}$$

En effet, dans notre cas, on utilise les notations $d_\star = d_k$ et $\lambda_\star = \lambda_{k+1}$.

En résumé, on a donné un équivalent de la condition d'optimalité de KKT du problème $(\mathcal{P}_{EI})$ puis on a linéarisé dans ce dernier F en z. Le problème alors obtenu est compliqué à résoudre mais on a remarqué qu'il est la condition d'optimalité de KKT du *problème quadratique osculateur*. Ainsi, si on résout ce-dernier, alors on résout également la condition d'optimalité de KKT du problème $(\mathcal{P}_{EI})$. Donc on obtiendra des points stationnaires du problème $(\mathcal{P}_{EI})$ en résolvant le *problème*

quadratique osculateur.

En pratique, on utilise une matrice définie positive M_k approximant H_k. Le problème quadratique osculateur est beaucoup plus facile à résoudre et on verra prochainement d'autres propriétés intéressantes sur M_k. Dans notre cas, on utilise la *factorisation de Cholesky modifiée* implémentée dans la fonction `cholmod`.

D'autre part, la résolution du problème OQS se fera à l'aide de la fonction `Quadprog`.

1.3.1 Mise à jour de l'optimiseur

Voici la mise à jour de l'optimiseur `sqp` :

```
function [x,lme,lmi,info] = sqp(simul,x,lme,lmi,options)
```

Paramètres d'entrée

- `simul` : simulateur utilisé dans l'optimiseur
- `x` : vecteur-colonne des abscisses puis des ordonnées des points d'articulation de la chaîne, $\mathtt{xy} = (\mathrm{x}_1, ..., \mathrm{x}_{n_n}, \mathrm{y}_1, ..., \mathrm{y}_{n_n})$. Il est de dimension $(2n_n, 1)$.
- `lme` : vecteur-colonne des multiplicateurs de Lagrange pour les contraintes d'égalité $\lambda_E = \{\lambda_i\}_{i=1}^{n_b}$. Il est de dimension $(n_b, 1)$.
- `lmi` : vecteur-colonne des multiplicateurs de Lagrange pour les contraintes d'inégalités $\lambda_I = \{\lambda_i\}_{i=n_b+1}^{n_b+n_n}$. Il est de dimension $(n_n, 1)$.

- `options` : structure spécifiant les paramètres de fonctionnement de l'algorithme, plus précisément, ils définissent le critère d'arrêt de l'algorithme de Newton.
  - `options.tol(1)`, `options.tol(2)` et `options.tol(3)` : les seuils de tolérance de $\|\nabla_x l(x_k, \lambda_k)\|_\infty$, $\|c(x_k)\|_\infty$ et $\|\min((\lambda_k)_I - c_I(x_k))\|_\infty$. Ils sont compris dans $]0, 1[$.
  - `options.maxit` : le nombre maximal d'itérations autorisées. C'est un entier positif.

Valeurs renvoyées

- `x`, `lme` et `lmi` : les vecteurs contenant les valeurs finales de $x_\star$, $(\lambda_\star)_E$ et $(\lambda_\star)_I$.
- `info` : structure donnant les informations sur le comportement de l'optimiseur suivantes :
  - `info.status` : décrit le résultat de l'algorithme.
    - 0 si le seuil d'optimalité est atteint ie l'algorithme a convergé.
    - 1 si les paramètres donnés à `sqp` ont un mauvais format.
    - 2 si le nombre maximal d'itérations autorisées est atteint ie il y a non convergence de l'algorithme.
  - `info.niter` : le nombre d'itérations effectuées.

1.4 Analyse théorique

1.4.1 Choix de matrice positive définie

Dans l'étude de la solution du problème quadratique osculateur, on a vu qu'on substitut la matrice Hessienne H_k par une matrice positive définie M_k. On fait ça car le problème devient plus

facile.

Naturellement on se pose alors la question de comment choisir cette matrice. On a vu (et implémenté) la factorisation de Cholesky modifiée pour cela, mais on peut se demander pourquoi ne pas simplement utiliser la matrice identité ?

On teste cette idée. Comme on verra dans le premier cas-test 4a plus en détails, la chaîne ci-dessous est la position finale de la chaîne dans un cas sans plancher, en utilisant Cholesky :

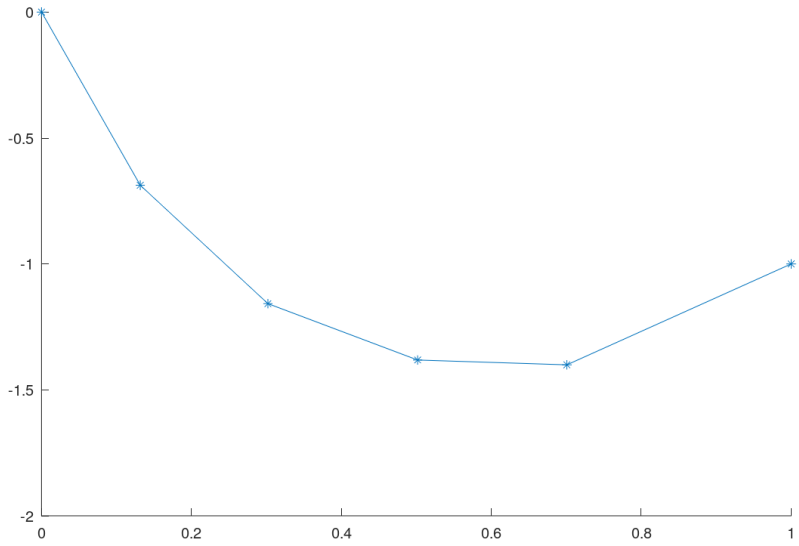


FIGURE 1 – Chaîne finale trouvé avec Cholesky pour un cas sans plancher

Puis en utilisant $M_k = Id$:

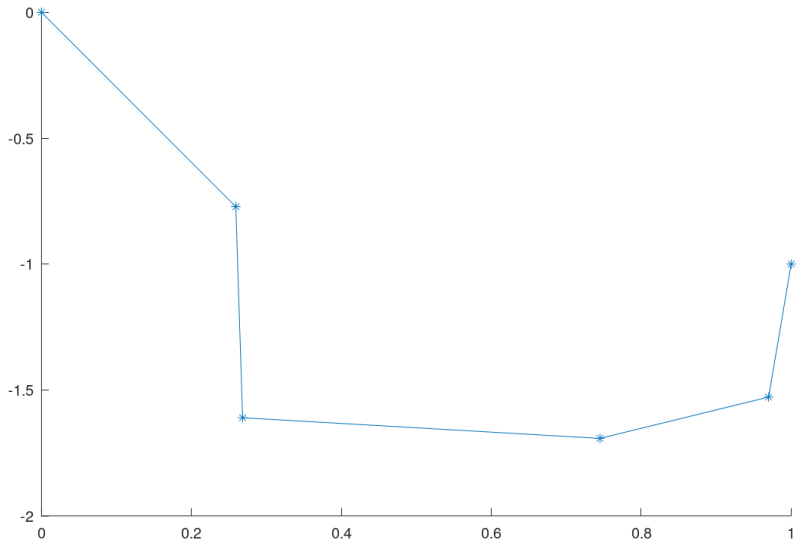


FIGURE 2 – Chaîne finale trouvé en substituant H_k par l'identité, cas sans plancher

Or, par des études précédentes, on sait que la première chaîne est un minimum du problème, et donc on s'approche bien de la bonne chaîne. On voit que la solution obtenue en utilisant l'identité ne semble pas du tout être la bonne chaîne. En fait, on réalise beaucoup plus d'itérations (on atteint `itermax`, alors il semble que l'algorithme ne converge pas), et les tolérances d'optimité ne diminuent pas trop (elles sont d'ordre 10^{-1}, lorsque pour Cholesky on obtient 10^{-5} en juste 18 itérations).

Ce qui arrive est qu'on peut mesurer une erreur de cette substitution. Si l'on substitut H_k par M_k, on écrit

$$H_k + E_k = M_k$$

où E_k est une matrice d'écart. On peut interpréter alors que plus les valeurs propres de E_k sont importantes, plus l'erreur est significative.

Sous cette interprétation, on la teste numériquement et on voit que dans le premier cas (Cholesky) les valeurs propres de E_k convergent toutes vers zéro (en fait, le code donne exactement zéro dans les dernières itérations). Pour l'identité ce n'est pas le cas : les dernières valeurs propres de E_k sont `(-1.58904, -1.58904, -0.62157, -0.62157, 0.35884, 0.35884, 0.67472, 0.67472)`. Ces valeurs sont assez grandes, et expliquent le mauvais comportement de la substitution par l'identité. On peut donc intuitivement se dire que l'identité approxime mal le hessien du lagrangien et que le convergence est donc très lente.

1.4.2 Choix d'un multiplicateur initial

Notre but maintenant est de proposer un multiplicateur initial plus efficace que $\lambda_0 = 0$, de telle sorte que si l'utilisateur donne au solveur un point initial x_0 qui est une « solution », l'algorithme

trouvera le multiplicateur optimal sans faire des solutions.

Or, si x_0 est une « solution », alors les conditions KKT du problème quadratique osculateur sont satisfaites pour ce x_0 et $d_0 = 0$ (spatialement on ne va pas bouger car on est déjà sur une solution). Alors la première équation du système KKT décrit plus haut devient

$$\nabla_x f(x_0) + c'(x_0)^\top \lambda_1 = 0 \qquad (*)$$

On propose alors d'utiliser cette équation pour définir le λ_1 initial, car dans le cas où on est déjà sur la solution on aura, sans faire aucune itération, le multiplicateur optimal.

Déterminer λ_1 dans l'équation $(*)$ n'est possible que si la matrice $c'(x_0)^\top$ est inversible à gauche. Or, le nombre de contraintes n est souvent supérieur au nombre de multiplicateurs m, donc $c'(x_0)^\top$ n'est pas inversible à gauche la plupart du temps. Pour pallier ce problème, on se ramène à un problème de minimisation par la fonction des moindres-carrés : on choisit λ_1 la solution du problème

$$\min_\lambda \|\nabla_x f(x_0) + c'(x_0)^\top \lambda_1\|_2^2$$

1.4.3 Critère pour minimalité locale

On cherche à trouver un critère permettant de déterminer si le point limite obtenu ne peut pas être un minimum local, en utilisant les conditions d'optimalité du second ordre pour les problèmes avec contraintes d'égalité.

On note $(\mathcal{P}_E)$ le problème avec seulement les contraintes d'égalités. On commence par montrer que si $(x_\star, \lambda_\star)$ est un point stationnaire de $(\mathcal{P}_E)$ vérifiant la condition nécessaire du second ordre pour les problèmes avec contraintes d'égalités et tel que $c'(x_\star)$ est surjective, alors $(x_\star, \lambda_\star)$ est dit régulier, c'est-à-dire que la matrice

$$\tilde{F} = \begin{pmatrix} \nabla^2_{xx}\ell(x_\star, \lambda_\star) & c'(x_\star)^\top \\ c'(x_\star) & 0 \end{pmatrix}$$

est inversible. Cette matrice est la matrice $F'(z_\star)$ avec F la fonction utilisée dans l'algorithme de Newton.

Preuve :

Pour montrer que $\tilde{F}$ est inversible, on va montrer qu'elle est injective. Soit $(d, \mu) \in \ker(\tilde{F})$. On suppose que la condition nécessaire du second ordre pour les problèmes avec contraintes d'égalités est vérifiée et que $c'(x_\star)$ est surjective. Alors

$$\begin{pmatrix} \nabla^2_{xx}\ell(x_\star, \lambda_\star) & c'(x_\star)^\top \\ c'(x_\star) & 0 \end{pmatrix} \begin{pmatrix} d \\ \mu \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\iff \begin{pmatrix} \nabla^2_{xx}\ell(x_\star, \lambda_\star)\, d \,+\, c'(x_\star)^\top \mu \\ c'(x_\star) d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Donc $d \in \ker(c'(x_\star))$ par la seconde ligne de l'égalité ci-dessus. Ensuite, on multiplie la première ligne par $d^\top$:

$$d^\top \nabla^2_{xx}\ell(x_\star, \lambda_\star)\, d \;+\; \underbrace{d^\top\, c'(x_\star)^\top}_{=(c'(x_\star)d)^\top=0}\; \mu = 0$$

Donc

$$d^\top \nabla^2_{xx}\ell(x_\star, \lambda_\star)\, d = 0$$

Or, par la condition CN2, on a

$$\nabla^2_{xx}\ell(x_\star, \lambda_\star) \succeq 0 \text{ sur } \ker(c'(x_\star))$$

Donc $d = 0$. Finalement, la première ligne de l'égalité matricielle devient

$$c'(x_\star)\, \mu = 0$$

Etant donné que $c'(x_\star)$ est surjective, on a $c'(x_\star)^\top$ injective et donc $\mu = 0$.

Finalement, on a montré que $\ker(F'(z_k)) = \{0\}$, c'est-à-dire que $F'(z_k)$ est inversible et le point $(x_\star, \lambda_\star)$ est bien régulier. □

D'autre part, on a que $c'(x_\star)$ est surjective (sous conditions).

Preuve :

On a $c'(x_\star) =$

$$\begin{pmatrix}
2(x_1-x_0) & 0 & \cdots & \cdots & \cdots & 0 & 2(y_1-y_0) & 0 & \cdots & \cdots & \cdots & 0 \\
2(x_1-x_2) & 2(x_2-x_1) & 0 & \cdots & \cdots & 0 & 2(y_1-y_2) & 2(y_2-y_1) & 0 & \cdots & \cdots & 0 \\
0 & 2(x_2-x_3) & 2(x_3-x_2) & 0 & \cdots & 0 & 0 & 2(y_2-y_3) & 2(y_3-y_2) & 0 & \cdots & 0 \\
\\
\\
0 & 2(x_2-x_3) & 2(x_3-x_2) & \cdots & 2(x_{N_b-2}-x_{N_b-1}) & 2(x_{N_b-1}-x_{N_b-2}) & 0 & \cdots & \cdots & 0 & 2(y_{N_b-2}-y_{N_b-1}) & 2(y_{N_b-1}-y_{N_b-2}) \\
0 & \cdots & \cdots & \cdots & 0 & 2(x_{N_b-1}-x_{N_b}) & 0 & \cdots & \cdots & \cdots & 0 & 2(y_{N_b-1}-y_{N_b})
\end{pmatrix}$$

Une autre formulation est

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

et

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

C'est une matrice $N_b \times 2N_n$. On veut donc montrer que N_b colonnes de $c'(x_\star)$ sont linéairement indépendantes pour en déduire qu'elle est surjective. On utilise le raisonnement suivant :

Sur la première ligne, on a 2 coefficients qui peuvent être non nuls. Plus précisément, on a :

$$x_1 = x_0 \quad \text{ou} \quad y_1 = y_0$$

et le « ou » de cette phrase est exclusif. En effet, si on a

$$x_1 = x_0 \quad \text{et} \quad y_1 = y_0$$

alors les points (x_0, y_0) et (x_1, y_1) sont confondus et donc $L_1 = 0$. Or, par hypothèse, les coefficients de L des longueurs finales des barres sont tous strictement non nuls.

On peut raisonner de la même manière sur les N_b lignes de $c'(x_\star)$. En effet, le raisonnement sur la dernière ligne est strictement le même et pour les lignes intermédiaires, on se ramène au même raisonnement en remarquant que les 4 termes de la ligne sont 2 couples d'opposés. Ainsi, sur chaque ligne, il existe au moins un coefficient non nul.

$\forall i \in \{1, ..., N_n\}$ la colonne associée à la colonne i est la colonne N_n+i. On a donc $\forall i \in \{1, ..., N_n\}$: la colonne i ou la colonne $N_n + i$ est non nulle (le « ou » est inclusif). Finalement, vu la « structure diagonale » de $c'(x_\star)$, on a au moins $N_n = N_b - 1$ colonnes indépendantes.

On atteint le point pathologique qui nous empêche d'affirmer que $c'(x_\star)$ est toujours surjective. En effet, il existe des cas où il n'y a que $N_b - 1$ colonnes indépendantes, c'est-à-dire que $\forall i \in \{1, ..., N_n\}$ la colonne i et sa colonne associée sont dépendantes.

On ajoute donc une condition pour montrer que $c'(x_\star)$ est surjective : il faut qu'au moins une colonne et sa colonne associée soient indépendantes. En effet, supposons que c'est le cas pour la colonne j. Alors $\forall i \in \{1, ..., N_n\} \setminus \{j\}$, les colonnes i et leur associée (ie $2(N_b - 2)$ colonnes) forment une famille de $N_b - 2$ colonnes indépendantes. Or on sait que les colonnes i et $N_n + i$ sont indépendantes entre elles et aussi avec les autres colonnes. Finalement, on a bien N_b colonnes indépendantes.

En résumé, si $\exists i \in \{1, ..., N_n\}$ tel que la colonne i et la colonne $N_n + i$ sont indépendantes, alors la matrice $c'(x_\star)$ est surjective. □

Ainsi, on arrive au critère recherché. Par contraposée, si le point $(x_\star, \lambda_\star)$ n'est pas régulier, alors $(x_\star, \lambda_\star)$ ne vérifie pas la CN2 ou $c'(x_\star)$ n'est pas surjective (le « ou » est inclusif). Or, on sait que $c'(x_\star)$ est surjective (sous conditions). Donc si le point $(x_\star, \lambda_\star)$ n'est pas régulier, alors $(x_\star, \lambda_\star)$ ne vérifie pas la CN2. C'est-à-dire, $x_\star$ ne peut pas être un minimum local du problème $(\mathcal{P}_E)$.

Si on rajoute les contraintes d'inégalités, on obtient 2 cas :
— si $x_\star$ vérifie les contraintes d'inégalités, alors $x_\star$ reste un point qui ne peut pas être un minimum local du problème $(\mathcal{P}_{EI})$.
— si $x_\star$ ne vérifie pas les contraintes d'inégalités, alors $x_\star$ n'est pas admissible et il est encore moins un minimum local du problème $(\mathcal{P}_{EI})$.

1.4.4 Observation d'un comportement différent

Comme le premier cas-test 4a montrera, il est possible que l'on observe un comportement différent de ce qui est arrivé précédemment avec l'algorithme de Newton (sans ou avec la recherche linéaire). Par exemple, on va regarder la convergence vers un minimum global du problème pour un cas test sans plancher, où précédemment on convergeait juste vers un point stationnaire.

La première chose que l'on peut se dire est que l'on résout un système $(\mathcal{P}_E)$ avec une méthode différente des précédents TPs. La méthode de Newton et cette nouvelle méthode trouve des points stationnaires. Ainsi, il est normal que l'on puisse aboutir sur des points stationnaires différents pour 2 méthodes différentes.

1.4.5 Vitesse de convergence

Finalement, on passe à l'étude de la vitesse de convergence de notre algorithme. On rappelle que pour trouver la vitesse α de convergence on cherche l'existence une constante $C > 0$ telle que

$$\frac{\|\text{critère}(x_{k+1})\|_\infty}{\|\text{critère}(x_k)\|_\infty^\alpha} \leq C \quad \forall k \in \mathbb{N}$$

Dans le cas $\alpha = 1$ on dit que la convergence est linéaire. Pour $1 < \alpha < 2$ on dit que la convergence est super-linéaire. Finalement, pour $\alpha = 2$, la convergence est quadratique. En pratique on fait un nombre fini d'itérations, alors on peut considérer que la constante n'existe pas si la valeur que l'on trouve est trop grande, et qu'elle existe si la valeur trouvée est relativement petite.

Dans des cas-tests, on va voir que notre algorithme possède ces trois comportements. Cependant, il est instructif de discuter déjà pourquoi on n'a pas la convergence quadratique dans quelques-uns. Cela arrive car on n'utilise pas la vraie Hessienne H_k du problème, mais une approximation M_k. Comme indiqué avant, utiliser la factorisation de Cholesky modifiée est un bon choix car on vérifie que les valeurs propres de la matrice d'écart E_k convergent vers zéro. C'est pour cela que même si l'on n'a pas une convergence quadratique, on conserve une convergence super-linéaire.

On note aussi l'existence du résultat de convergence locale suivant, dû à Bonnans : Si
— f et c sont de classe $C^{2,1}$ dans un voisinage d'un minimum local x_* de (P_{EI}),
— il existe un *unique* multiplicateur optimal $\lambda_* \in \mathbb{R}^m$ associé à x_*,
— les conditions suffisantes d'optimalité du second ordre ont lieu

alors il existe un voisinage V de (x_*, λ_*) tel que si le premier itéré $(x_1, \lambda_1) \in V$,
— l'algorithme OQS (que l'on utilise pour résoudre le problème quadratique osculateur) est bien défini (il *peut* calculer une suite d'itérés $\{(x_k, \lambda_k)\}$),
— la convergence de la suite $\{(x_k, \lambda_k)\}$ vers $(x_\star, \lambda_\star)$ est quadratique.

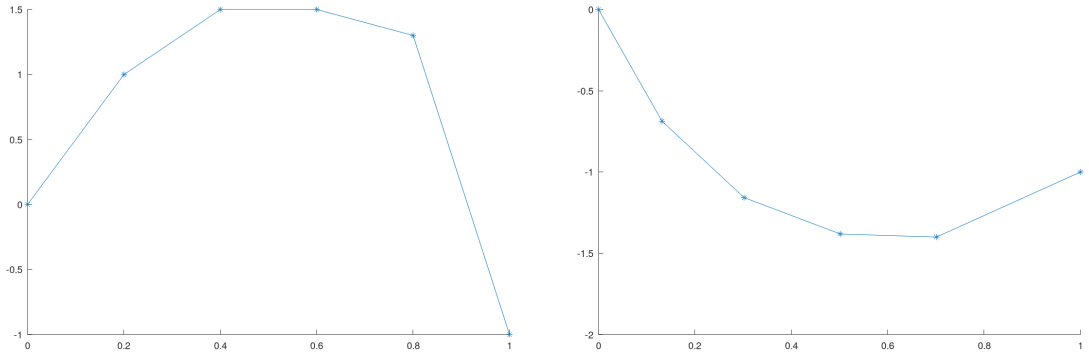
Ainsi, on peut donner d'autres explications sur pourquoi la convergence n'est pas quadratique : la non-unicité du multiplicateur optimal sur $x_\star$, ou alors la non-vérification des conditions suffisantes d'optimalité du second ordre.

1.5 Étude des cas test

1.5.1 Cas-test 4.a

Le premier cas test utilise les données suivantes : on prendra des barres de longueur final `L = [0.7 0.5 0.3 0.2 et 0.5]`. Le deuxième point de fixation de la chaîne est $(a, b) = (1, -1)$, et la position initiale des noeuds est donnée par `xy = [ 0.2 0.4 0.6 0.8 ... 1 1.5 1.5 1.3]`.

Notons que ce cas n'a pas de plancher, donc c'est juste un test de l'algorithme dans un cas avec contraintes d'égalité. On représente ci-dessous à gauche le graphe initial de la chaîne, et à droite le graphe de la chaîne dans sa position finale.



Les **valeurs finales des indicateurs d'optimalité** sont $1.935792e-05$ (gradient) , $3.109633e-06$ (contraintes), et on n'a pas la troisième car ce cas n'a pas de plancher. Le nombre maximal d'itérations autorisées est 500 et le **nombre d'itérations effectuées** est 18.

On trouve que la **vitesse de convergence est constante** ($\alpha = 0$), avec constante $C = 1.22168$.

Les **points finaux de la chaîne** sont : `[0.131696 0.301985 0.501695 0.700783 -0.6875 -1.15761 -1.38148 -1.40059]`.

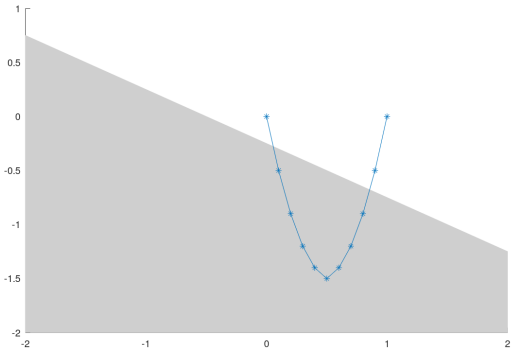
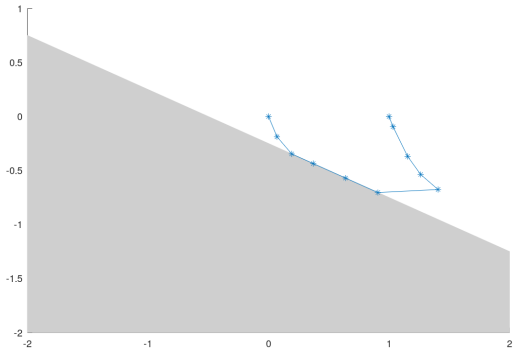
Les **multiplicateurs finaux (contraintes d'égalité)** de la chaîne sont `[0.926125 0.716241 0.610674 0.612613 0.407621]`.

Les **multiplicateurs finaux (contraintes d'inégalité)** de la chaîne sont `[]`, c'est-à-dire, on n'a pas de multiplicateurs d'inégalités, car il n'y a pas des contraintes d'inégalités.

1.5.2 Cas-test 4.b

Le deuxième cas test utilise les données suivantes : on prendra des barres de longueur final `L = [0.2 0.2 0.2 0.3 0.3 0.5 0.2 0.2 0.3 0.1]`. Le deuxième point de fixation de la chaîne est $(a, b) = (1, 0)$, et la position initiale des noeuds est donnée par `xy = [ 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 -0.5 -0.9 -1.2 -1.4 -1.5 -1.4 -1.2 -0.9 -0.5]`.

On décrit un seul plancher, avec `R = -0.25; S = -0.5;`

Les **valeurs finales des indicateurs d'optimalité** sont $8.391198e-04$ (gradient) , $1.652902e-07$ (contraintes), et $4.467909e-08$ (minimum entre λ_I et $-c_I$). Le nombre maximal d'itérations autorisées est 500 et le **nombre d'itérations effectuées** est 14.

On trouve que la **vitesse de convergence est super-linéaire** ($\alpha = 1.5$), avec constante $C = 13.5232$.

Les **points finaux de la chaîne** sont : `[0.0701768 0.191921 0.370807 0.639135 0.907463 1.40667 1.26267 1.15409 1.0331 -0.187284 -0.345961 -0.435403 -0.569567 -0.703732 -0.675638 -0.536841 -0.368883 -0.0943619]`.

Les **multiplicateurs finaux (contraintes d'égalité)** de la chaîne sont `[1.0436 0.601542 0.317344 0.0252236 -0.198479 -0.343627 1.19128 1.57984 1.4213 5.19465 ]`.

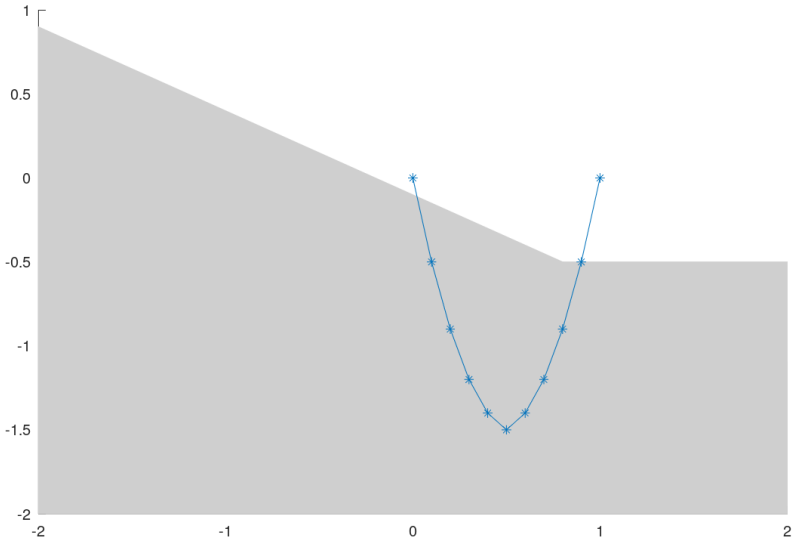
Les **multiplicateurs finaux (contraintes d'inégalité)** de la chaîne sont `[0 0.0658683 0.2 0.24 0.472679 0 0 0 0 ]`.

1.5.3 Cas-test 4.c

Le troisième cas test utilise exactement les mêmes données précédentes : on prendra des barres de longueur final `L = [0.2 0.2 0.2 0.3 0.3 0.5 0.2 0.2 0.3 0.1]`. Le deuxième point de fixation de la chaîne est $(a, b) = (1, 0)$, et la position initiale des noeuds est donnée par `xy = [0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 -0.5 -0.9 -1.2 -1.4 -1.5 -1.4 -1.2 -0.9 -0.5]`.

Ce qui change est que maintenant on décrit un plancher en utilisant deux fonctions affines, avec `R = [-0.25; -0.5]` et `S = [-0.5; 0]`

Voici la figure qui représente la chaîne dans sa position initiale :

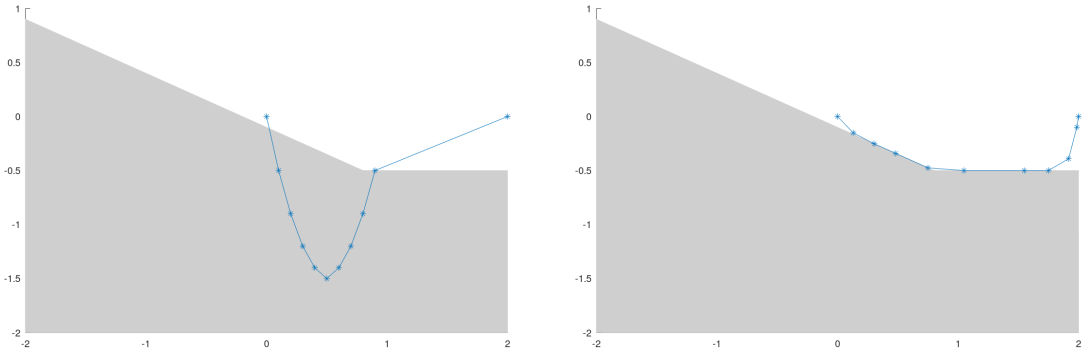


Or, notre algorithme indique que dans ce cas, le problème quadratique osculateur n'a pas de solution. Ainsi, cette même figure représente la chaîne finale, car on ne fait aucune itération.

Les points finaux de la chaîne sont les mêmes que les points initiaux.
Les multiplicateurs finaux sont les mêmes que les initiaux, mais on les a choisit arbitrairement, donc peu importe ses valeurs.

1.5.4 Un cas-test extra

Faisons une petite modification pour vérifier que le fait que le cas précédent n'ait pas de solution n'est pas un bug de notre code, mais en fait un fait du problème. On modifie juste la coordonnée x du deuxième point extrémal : $(a, b) = (2, 0)$. Or, dans ce cas l'algorithme converge et voici les chaîne obtenues :



Les **valeurs finales des indicateurs d'optimalité** sont $1.749183e-05$ (gradient) , $5.383627e-07$ (contraintes), et $5.551115e-17$ (minimum entre λ_I et $-c_I$). Le nombre maximal d'itérations autori-

sées est 500 et le **nombre d'itérations effectuées** est 24.

On trouve que la **vitesse de convergence est linéaire** ($\alpha = 1$), avec constante $C = 24.7395$.

Les **points finaux de la chaîne** sont : `[0.130709 0.303654 0.482539 0.750868 1.04986 1.54986 1.74986 1.91762 1.98615 -0.151378 -0.251827 -0.34127 -0.475434 -0.5 -0.5 -0.5 -0.391104 -0.0990366 ]`.

Les **multiplicateurs finaux (contraintes d'égalité)** de la chaîne sont `[1.3252 1.00156 0.775998 0.330993 0.115713 0.0691943 0.172986 0.206199 0.504848 2.49854 ]`.

Les **multiplicateurs finaux (contraintes d'inégalité)** de la chaîne sont `[0 0.137603 0.2 0.216871 0 0 0 0 0 0 0 0 0 0.394315 0.35 0.155098 0 0 ]`.

2 Version quasi-newtonienne

2.1 Méthode de quasi-Newton

Lors de travaux précédents, on a utilisé la méthode de Newton appliquée à un problème avec contraintes d'égalités. L'objectif était de trouver un point stationnaire du problème. Afin d'introduire la méthode de quasi-Newton que nous allons utiliser pour résoudre notre problème avec contraintes d'égalités et d'inégalités, on simplifie la théorie en considérant un problème sans contraintes

$$\begin{cases} \min f(x) \\ x \in \mathbb{R}^n \end{cases} \tag{1}$$

On introduit les notations suivantes :
— $(x_k)_k$ la suite des itérés
— $g_k = \nabla f(x_k)$
— d_k la direction de descente et $\alpha_k > 0$ le pas permettant de construire l'itéré $k+1$:

$$x_{k+1} = x_k + \alpha_k d_k$$

Dans la méthode de Newton, on détermine la direction de descente avec

$$d_k = -\nabla^2 f(x_k)^{-1} g_k$$

Une méthode de quasi-Newton se base sur le même principe où on remplace $\nabla^2 f(x_k)$ par une matrice M_k ayant de bonnes propriétés que l'on va découvrir. Elle est notamment symétrique définie positive.

Le remplacement de $\nabla^2 f(x_k)$ est motivé par de nombreux points :
— Il n'est pas obligé de calculer les dérivées secondes des fonctions intervenant dans la définition du problème d'optimisation à résoudre.
— Le problème quadratique osculateur (PQO) est toujours « mieux » posé (il a au plus une solution).

— La fonction f peut ne pas être 2 fois dérivables.
— On veut obtenir une convergence en utilisant uniquement les dérivées de premier ordre, quitte à avoir une convergence moins rapide.

A partir de maintenant, on se place dans le cadre de l'étude du problème et on pose $f(\cdot) = \ell(\cdot, \lambda) \ \ \forall \lambda \in \mathbb{R}$.

Dans la méthode de quasi-Newton, on utilise la variation du gradient de $\ell(\cdot, \lambda)$ afin d'approximer le hessienne $\nabla^2 \ell(\cdot, \lambda)$. D'un point de vue de l'itération k, on veut construire M_{k+1} telle que

$$M_{k+1} \approx \nabla^2_{xx} \ell(x_k, \lambda_{k+1})$$

où on remarque que l'on utilise λ_{k+1} et non λ_k car, à l'itération k, on calcule (x_{k+1}, λ_{k+1}) avant M_{k+1}.

On introduit deux nouvelles notations :
— l'écart en position $\delta_k = x_{k+1} - x_k$
— le reste intégral du développement de Taylor du lagrangien

$$\gamma_k = \underbrace{\left(\int_0^1 \nabla^2 \ell(x_k + t\delta_k, \lambda_{k+1}) dt \right)}_{\text{« hessienne moyenne »}} \delta_k$$

On peut l'approximer par l'écart en gradient du lagrangien $\gamma_k \approx g_{k+1} - g_k = \nabla \ell(x_{k+1}, \lambda_{k+1}) - \nabla \ell(x_k, \lambda_{k+1})$.

Par intuition, on veut que M_{k+1} vérifie l'égalité en remplacement de la *hessienne moyenne* pour satisfaire l'objectif d'approximation de $\nabla^2 \ell(\cdot, \lambda)$. On appelle cette égalité l'*équation de quasi-Newton* et on la considère comme une contrainte dans la détermination de M_{k+1} :

$$\gamma_k = M_{k+1} \delta_k$$

On verra que l'on utilisera cette équation dans l'analyse théorique de cette méthode, notamment pour déterminer un bon itéré initial M_1 (voir la partie 2.2.4).

La seconde contrainte sur M_{k+1} est la symétrie : $M_{k+1} = M_{k+1}^\top$. En effet, la symétrie étant une propriété du hessien du lagrangien, on souhaite que les matrices M_k soient elles-aussi symétriques.

Ensuite, on souhaite que la suite des matrices $(M_k)_k$ soit stable dans le sens où on va chercher M_{k+1} telle que $M_k^{-1/2} M_{k+1} M_k^{-1/2}$ soit le plus proche possible de l'identité I. On va étudier la minimisation de ce critère abstrait à travers la minimisation d'un autre critère ψ. On pose alors M_{k+1} la solution du problème suivant :

$$\begin{cases} \min_M \psi(M, M_k) \\ \gamma_k = M \delta_k \\ M = M^\top \end{cases} \tag{2}$$

On pourrait ajouter à ce problème une dernière contrainte : les matrices M_k sont définies positives. En effet, premièrement, les matrices M_k doivent approcher $\nabla^2 \ell(\cdot, \lambda)$ qui doit être semi-définie positive en une solution d'après la condition nécessaire d'optimalité d'ordre 2 pour les problèmes sans contraintes. Deuxièmement, si les matrices M_k sont définies positives, alors la direction associée aux méthodes de quasi-Newton définie par

$$d_k = -M_k^{-1} g_k$$

est bien une direction de descente pour le problème (1). En effet, la propriété de définie positivité est stable par passage à l'inverse et alors

$$\langle d_k, g_k \rangle = -\langle M_k^{-1} g_k, g_k \rangle < 0$$

car $g_k \neq 0$ (sinon l'algorithme se serait arrêté à l'itération précédente).

On peut alors ajouter cette contrainte au problème (2). Cependant, l'ajouter de manière explicite peut être problématique :

— L'ensemble des matrices symétriques définies positives $\mathcal{S}_{++}^n$ est un ouvert de $\mathcal{M}_n(\mathbb{R})$. Ainsi, le problème (2) peut ne pas avoir de solutions. On ne peut notamment pas utiliser le théorème de Weierstrass.

— Si on demande à ce que M_k appartienne à l'ensemble des matrices symétriques semi-définies positives, alors elle n'est pas forcément inversible. Ainsi, si c'est le cas, la direction de descente $d_k = -M_k^{-1} g_k$ n'est pas définie.

On va donc ajouter la contrainte de définie positivité des M_k de manière implicite dans la fonction-objectif. C'est une forme de pénalisation. On cherche donc une fonction-objectif ψ définie sur les matrices symétriques respectant l'équation de quasi-Newton et telle que si ces matrices ne sont pas définies positives, alors ψ vaudra $+\infty$. On pose

$$\begin{aligned} \psi : \mathcal{S}^n &\to \mathbb{R} \cup \{+\infty\} \\ M &\mapsto \mathrm{tr} M + \mathrm{ld} M \end{aligned}$$

où $\mathrm{ld} M$ est la fonction log-déterminant définie par

$$\begin{aligned} \mathrm{ld} : \mathcal{S}^n &\to \mathbb{R} \cup \{+\infty\} \\ M &\mapsto \begin{cases} -\log\det M & \text{si } M \in \mathcal{S}_{++}^n \\ +\infty & \text{sinon} \end{cases} \end{aligned}$$

Etant donné que $\mathrm{tr} M = \sum_i \lambda_i$ et $\det M = \Pi_i \lambda_i$, on peut reformuler $\psi(M)$ si $M \in \mathcal{S}_{++}^n$:

$$\psi(M) = \sum_{i=1}^{n} (\lambda_i - \log \lambda_i) \quad \text{si } M \in \mathcal{S}_{++}^n$$

On a que l'une des valeurs propres de M tend vers 0 ou vers $+\infty$ ssi $\psi \to +\infty$. Autrement dit, le domaine de ψ est $\mathcal{S}_{++}^n$. Intuitivement, si $M \in \mathcal{S}_{++}^n$ se rapproche du bord de $\mathcal{S}_{++}^n$, alors $\psi \to \infty$, c'est-à-dire que M ne peut pas être solution de (2). On obtient donc bien la contrainte implicite de

définie positivité recherchée.

Une autre propriété intéressante de la fonction ψ est que son unique minimiseur est la matrice identité (car chaque fonction $\lambda_i \mapsto \lambda_i - \log \lambda_i$ prend son minimum lorsque $\lambda_i = 1$ et l'unique matrice carrée de taille n ayant n valeurs propres égales à 1 est l'identité). Or, le critère que l'on souhaite appliquer à la suite des matrices $(M_k)_k$ est « construire $(M_k)_k$ telle que $M_k^{-1/2} M M_k^{-1/2}$ soit le plus proche possible de l'identité I ». Donc on peut se ramener au problème de minimisation suivant

$$\begin{cases} \min_M \psi(M_k^{-1/2} M M_k^{-1/2}) \\ \gamma_k = M\delta_k \\ M \in \mathcal{S} \end{cases}$$

et étant donné que la contrainte $M \in \mathcal{S}_{++}^n$ est implicite, on a finalement

$$\begin{cases} \min_M \psi(M_k^{-1/2} M M_k^{-1/2}) \\ \gamma_k = M\delta_k \\ M \in \mathcal{S}_{++}^n \end{cases} \tag{3}$$

On étudie ensuite le cas $\delta_k = 0$:
— si $\gamma_k \neq 0$, alors le problème (3) n'a pas de solution à cause de la contrainte $\gamma_k = M\delta_k$.
— si $\gamma_k = 0$, alors le problème (3) devient

$$\begin{cases} \min_M \psi(M_k^{-1/2} M M_k^{-1/2}) \\ M \in \mathcal{S}_{++}^n \end{cases}$$

et sa solution est M_k car on sait que ψ atteint son minimum en l'identité I.

On s'intéresse finalement au cas qui nous intéresse : celui où $\delta_k \neq 0$. On peut alors montrer les deux points suivants :
— Le problème (3) a une solution ssi $\gamma_k^\top \delta_k > 0$. On s'intéresse à ceci dans la partie 2.2.3.
— La solution M_{k+1} du problème (3) est unique et est donnée par

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

Cette formule est appelée *formule de BFGS*.

On voit qu'il est nécessaire que $\gamma_k^\top \delta_k$ soit strictement positif pour que le problème (3) ait une solution. On a d'ailleurs défini γ_k comme étant l'approximation de l'écart du hessien du lagrangien $\nabla^2_{xx} \ell(\cdot, \lambda)$. On pourrait alors se dire que $\gamma_k^\ell := \nabla \ell(x_{k+1}, \lambda_{k+1}) - \nabla \ell(x_k, \lambda_{k+1})$ est un bon choix de γ_k vérifiant $\gamma_k^\top \delta_k > 0$. Cependant, on verra dans la partie 2.2.3 que cela n'est pas garanti.

Pour résoudre ce problème éventuel, on introduit la *correction de Powell* qui consiste à prendre

$$\gamma_k = (1 - \theta) M_k \delta_k + \theta \gamma_k^\ell$$

où $\theta \in]0, 1]$ est le plus proche de 1 tel que

$$\gamma_k^\top \delta_k \geq 0.2\delta_k^\top M_k \delta_k$$

γ_k est donc une combinaison convexe où l'un des deux membres est l'écart des lagrangiens γ_k^ℓ. On souhaite avoir θ le plus proche de 1 car alors on se ramène à $\gamma_k = \gamma_k^\ell$ et on aura $\gamma_k^\top \delta_k > 0$. De plus, on peut intuitivement se dire que plus θ est proche de 1, plus on prend de l'information sur le hessien du lagrangien. A l'inverse, si θ est proche de 0, alors on a $\gamma_k = M_k\delta_k \Longrightarrow \delta_k^\top \gamma_k = \delta_k^\top M_k \delta_k > 0$ car M_k est définie positive. Le point positif de prendre $\theta = 0$ est qu'on a la condition nécessaire $\gamma_k^\top \delta_k > 0$ vérifiée. Le problème ici est que la suite des matrices $(M_k)_k$ va être constante et la convergence de l'algorithme risque d'être très lente.

On peut montrer que θ est déterminé par

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

La dernière étape à déterminer pour construire le nouvel algorithme est son initialisation. On prendra $M_1 = I$ afin de calculer x_2. Ensuite, on pose $M_1 = \eta_1 I$ où η_1 « reflète l'échelle du problème ». On étudie ce choix de η_1 dans la partie 2.2.4. Après cela, on peut calculer les autres matrices M_k avec $k \geq 2$ par la formule de BFGS.

2.2 Analyse théorique

2.2.1 Caractère bien posé du PQO

La première chose à vérifier lorsqu'on pose un problème est l'existence de solution. Dans le PQO, il peut y avoir une incompatibilité des contraintes du problème

$$\begin{cases} \min_{d\in\mathbb{R}^n} \nabla f(x_k)^\top d + \frac{1}{2} d^\top M_k d \\ c_E(x_k) + c_E'(x_k)d = 0 \\ c_I(x_k) + c_I'(x_k)d \leq 0 \end{cases}$$

Si les contraintes sont incompatibles, alors il n'existe pas de solution.

Les contraintes d'égalités forment une intersection de droites qui peut ne pas être inclue dans l'intersection des demi-plans formée par les contraintes d'inégalités. On peut prendre l'exemple simple de $\mathbb{R}^2$ composé de la contrainte d'égalité $y = -2$ et de la contrainte d'inégalité $y \geq 0$. Ces contraintes sont incompatibles.

2.2.2 Approximation du hessien du lagrangien par la formule de BFGS

Il peut être contradictoire d'approcher le hessien du lagrangien $\nabla^2_{xx}\ell(x,\lambda)$ par une matrice M définie positive car le hessien du lagrangien peut ne pas être défini positif.

Par exemple, on pose le problème avec contraintes d'égalités suivant :

$$\begin{cases} \min_x -x^2 \\ x = 0 \end{cases}$$

Le lagrangien est $\ell(x,\lambda) = -x^2$ et son hessien est $\nabla^2_{xx}\ell(x,\lambda) = -2$ qui n'est pas défini positif.

2.2.3 Sur le choix de γ_k dans la formule de BFGS

Le vecteur γ_k devrait idéalement être γ_k^ℓ. Idéalement, on a défini γ_k par

$$\gamma_k = \underbrace{\left(\int_0^1 \nabla^2 \ell(x_k + t\delta_k, \lambda_{k+1}) dt \right)}_{\text{« hessienne moyenne »}} \delta_k$$

C'est le reste intégral du développement de Taylor du lagrangien $\ell(\cdot, \lambda)$. Autrement dit

$$\nabla \ell(x_{k+1}, \lambda_{k+1}) - \nabla \ell(x_k, \lambda_{k+1}) = \gamma_k$$

Cependant, en pratique, on ne connaît pas la hessienne mobile et donc on n'a pas l'égalité. γ_k est donc une approximation de l'écart du hessien du lagrangien $\nabla^2_{xx} \ell(\cdot, \lambda)$. On peut alors se dire que $\gamma_k^\ell := \nabla \ell(x_{k+1}, \lambda_{k+1}) - \nabla \ell(x_k, \lambda_{k+1})$ est un bon choix de γ_k.

On doit avoir $\gamma_k^\top \delta_k > 0$ pour que M_{k+1} soit définie positive. On prouve ceci par contradiction. Supposons que $\gamma_k^\top \delta_k \leq 0$. La matrice M_{k+1} étant solution du problème (3), elle doit respecter la contrainte constituée de l'égalité de quasi-Newton :

$$\gamma_k = M_{k+1} \delta_k$$

On la transpose :

$$\gamma_k^\top = \delta_k^\top M_{k+1}^\top$$

Or la matrice M_{k+1} est symétrique par contrainte du problème (3). Donc

$$\gamma_k^\top = \delta_k^\top M_{k+1}$$

Enfin, on multiplie par δ_k :

$$\gamma_k^\top \delta_k = \delta_k^\top M_{k+1} \delta_k$$

Par hypothèse, on a $\gamma_k^\top \delta_k \leq 0$, donc $\delta_k^\top M_{k+1} \delta_k \leq 0$. C'est-à-dire que M_{k+1} n'est pas définie positive. On a donc bien montré que l'on doit avoir $\gamma_k^\top \delta_k > 0$ pour que M_{k+1} soit définie positive.

Le choix $\gamma_k = \gamma_k^\ell$ ne garantit pas la définie positivité de M_{k+1}. Le point précédent nous donne une condition nécessaire sur le choix de γ_k pour que M_{k+1} soit définie positive. On cherche donc à montrer que le choix $\gamma_k = \gamma_k^\ell$ n'implique pas forcément $\gamma_k^\top \delta_k > 0$ et que donc le choix $\gamma_k = \gamma_k^\ell$ ne garantit pas la définie positivité de M_{k+1}.

On a $\gamma_k^\ell = \nabla \ell(x_{k+1}, \lambda_{k+1}) - \nabla \ell(x_k, \lambda_{k+1})$. C'est une intégration du hessien entre x_{k+1} et x_k.

Supposons qu'on se situe dans un problème avec contraintes d'égalités. Le hessien du lagrangien $\nabla^2_{xx} \ell(x, \lambda)$ peut avoir une courbure négative transversalement à ces contraintes d'égalités. En effet, la condition nécessaire d'ordre 2 pour les problèmes avec contraintes d'égalités implique que

$$\nabla^2_{xx} \ell(x, \lambda) \succeq 0 \quad \text{sur } N(c'(x))$$

mais rien ne nous indique que c'est toujours le cas en dehors de $N(c'(x))$. Donc, avec le choix $\gamma_k = \gamma_k^\ell$, on peut ne pas avoir $\gamma_k^\top \delta_k > 0$ et donc ce choix ne garantit pas la définie positivité de M_{k+1}.

La correction de Powell est préférable. Résumons les 3 points précédents. On a une condition nécessaire sur γ_k pour que la matrice M_{k+1} soit définie positive. C'est l'inégalité $\gamma_k^\top \delta_k > 0$. D'autre part, on a vu qu'on candidat idéal de γ_k est γ_k^ℓ. Cependant, on a aussi vu que γ_k^ℓ ne vérifie pas forcément la condition nécessaire et que donc on ne peut pas le choisir. On souhaite donc trouver une manière de calculer γ_k pour qu'il satisfasse la condition nécessaire et qu'il se rapproche le plus possible de γ_k^ℓ. C'est pourquoi on utilise la *correction de Powell* qui consiste à prendre

$$\gamma_k = (1-\theta) M_k \delta_k + \theta \gamma_k^\ell$$

où $\theta \in]0, 1]$ est le plus proche de 1 tel que

$$\gamma_k^\top \delta_k \geq 0.2 \delta_k^\top M_k \delta_k$$

Etant donné que la matrice M_k est définie positive, on a $\delta_k^\top M_k \delta_k > 0$ et donc γ_k vérifie la condition nécessaire $\gamma_k^\top \delta_k > 0$.

De plus, dans la correction de Powell, on cherche à avoir θ le plus proche de 1. Ainsi, on se ramène le plus possible au cas idéal $\gamma_k = \gamma_k^\ell$. C'est le cas privilégié par la correction de Powell et on a $\gamma_k \neq \gamma_k^\ell$ uniquement si la condition nécessaire n'est pas vérifiée.

Finalement, la correction de Powell vérifie bien les 2 propriétés recherchées :
— γ_k vérifie la condition nécessaire $\gamma_k^\top \delta_k > 0$
— γ_k se rapproche le plus possible de γ_k^ℓ

2.2.4 Initialisation de M_1 par $\eta_1 I$.

Avant d'initialiser l'algorithme, on dispose de très peu d'informations. On pose alors $M_1 = I$ car l'identité est bien symétrique définie positive. Cela nous permet de calculer x_2. La prochaine étape consiste à calculer la matrice M_2 à partir de la formule de BFGS et on construira les matrices M_k suivantes avec cette même formule. Cependant, calculer M_2 à partir de $M_1 = I$ n'est pas optimal d'un point de vue de la convergence. En effet, l'identité peut être éloigné du hessien du lagrangien. il faudra alors beaucoup d'itérations pour que la suite des matrices $(M_k)_k$ approchent correctement le hessien du lagrangien.

Une manière de corriger cela est de « mettre à l'échelle » l'identité en remplaçant $M_1 = I$ par $M_1 = \eta_1 I$ où η_1 est un réel reflétant l'échelle du problème. On défini η_1 par

$$\eta_1 = \frac{\gamma_1^\top \gamma_1}{\gamma_1^\top \delta_1}$$

que l'on peut calculé car on connaît la valeur de x_2 à ce stade de l'algorithme.

« η_1 reflète l'échelle du problème » si $M_1 = \eta_1 I$ réalise l'équation de quasi-Newton dans la direction γ_1. Intuitivement, on veut que la matrice initiale M_1 vérifie les contraintes du problème

(3), d'où la vérification de quasi-Newton (la symétrie est acquise car $M_1 = \eta_1 I$ l'est pour un η_1 quelconque). Plus précisément, l'équation de quasi-Newton est vérifiée dans la direction γ_1 car c'est l'écart en les gradients des lagrangiens initiaux (et c'est d'ailleurs la seule source d'informations sur le hessien du lagrangien à l'itération initiale).

On doit donc montrer que

$$\gamma_1^\top \gamma_1 = \gamma_1^\top M_1 \delta_1$$

On a

$$\gamma_1^\top M_1 \delta_1 = \gamma_1^\top \eta_1 I \delta_1 = \gamma_1^\top \underbrace{\frac{\gamma_1^\top \gamma_1}{\gamma_1^\top \delta_1}}_{\in \mathbb{R}} \delta_1$$

On peut donc permuter $\gamma_1^\top$ et $\frac{\gamma_1^\top \gamma_1}{\gamma_1^\top \delta_1}$:

$$\begin{aligned}
\gamma_1^\top M_1 \delta_1 &= \gamma_1^\top \frac{\gamma_1^\top \gamma_1}{\gamma_1^\top \delta_1} \delta_1 \\
&= \frac{\gamma_1^\top \gamma_1}{\gamma_1^\top \delta_1} \gamma_1^\top \delta_1 \\
&= \gamma_1^\top \gamma_1
\end{aligned}$$

2.3 Implémentation du code

On reprend le code en ajoutant la possibilité de résoudre le problème osculateur avec les matrices M_k générées par la formule de BFGS. Pour choisir la méthode de résolution, on ajoute une nouvelle option : `option.deriv`. Si elle vaut 1, on utilise la méthode de quasi-Newton (BFGS) et si elle vaut 2, on utilise la méthode de Newton.

Lors de la mise à jour des itérés x_{k+1} et λ_{k+1}, on utilise un pas α_k que l'on calcule par la règle de Wolf. Cette règle complète celle d'Armijo en ajoutant une contrainte supplémentaire, empêchant le pas d'être trop petit.

Premièrement, comme dans la règle d'Armijo, le pas α_k doit vérifier l'inégalité suivante :

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \omega_1 \alpha_k \langle g_k, d_k \rangle$$

De plus, il vérifie l'inégalité suivante :

$$\langle \nabla f(x_k + \alpha_k d_k), d_k \rangle \geq \omega_2 \langle g_k, d_k \rangle$$

Les constantes ω_1 et ω_2 vérifient

$$0 < \omega_1 < \omega_2 < 1$$

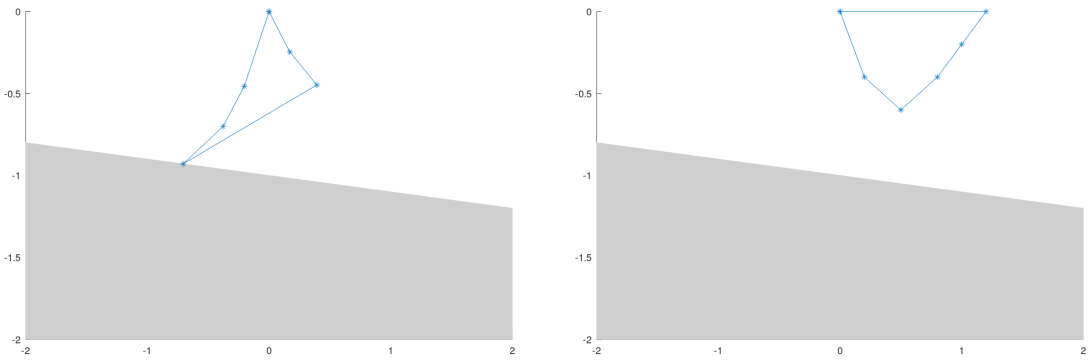
Dans notre code, nous avons utilisé $\omega_1 = 10^{-4}$ et $\omega_2 = 0.99$.

2.4 Étude des cas-test

2.4.1 Cas-test 5.a

Le premier cas test utilise les données suivantes : on prend 6 barres de longueurs finales `L = [0.5 0.3 0.4 1.2 0.3 0.3]`. Le deuxième point de fixation de la chaîne est $(a, b) = (0, 0)$, et la position initiale des noeuds est donnée par `xy = [ 0.2 0.5 0.8 1.0 1.2 -0.4 -0.6 -0.4 -0.2 0. ]`.

On décrit un seul plancher, avec `R = -1; S = -0.1;`



Les **valeurs finales des indicateurs d'optimalité** sont $1.332401e-04$ (gradient) , $1.096647e-08$ (contraintes), et $1.110223e-16$ (minimum entre λ_I et $-c_I$). Le nombre maximal d'itérations autorisées est 300 et le **nombre d'itérations effectuées** est 19.

Les **points finaux de la chaîne** sont : `[-0.204342 -0.377671 -0.706204 0.393064 0.170793 -0.456338 -0.701199 -0.92938 -0.44812 -0.246637 ]`

Les **multiplicateurs finaux (contraintes d'égalité)** de la chaîne sont `[1.19256 1.40588 0.741686 -0.25373 1.25491 1.63354 ]`

Les **multiplicateurs finaux (contraintes d'inégalité)** de la chaîne sont `[0 0 0.705767 0 0 ]`.

Voici un tableau avec toutes les informations sur le déroulement de l'algorithme :

iter	|gl|	|ce|	(ci, lmi)	|x|	|lm|	Powell	cond(M)
1	7.81e+01	8.08e+00	2.22e-16	2.7e+00	1.11e+01	4.1e-02	1.0e+00
2	2.75e+00	2.01e+00	1.11e-16	1.6e+00	1.28e+00	1.0e+00	2.7e+00
3	8.09e-01	6.04e-01	0.00e+00	1.2e+00	5.32e-01	1.0e+00	8.5e+00
4	4.16e-01	2.79e-01	0.00e+00	1.1e+00	1.92e+00	1.0e+00	9.2e+00
5	4.71e+00	4.59e-01	2.22e-16	1.0e+00	2.83e+00	1.0e+00	1.1e+01
6	1.24e+00	1.02e-01	0.00e+00	1.0e+00	2.07e+00	1.0e+00	8.6e+00
7	2.27e-01	2.15e-02	0.00e+00	1.0e+00	1.87e+00	1.0e+00	7.7e+00
8	4.88e-02	1.93e-03	0.00e+00	1.0e+00	1.86e+00	1.0e+00	8.7e+00
9	5.33e-02	4.78e-04	0.00e+00	1.0e+00	1.87e+00	7.8e-01	3.9e+01
10	1.52e-01	4.19e-02	0.00e+00	1.0e+00	1.88e+00	3.5e-01	9.6e+02
11	7.46e-01	7.74e-01	1.11e-16	9.3e-01	1.70e+00	8.1e-01	2.0e+04
12	8.25e-01	6.77e-02	0.00e+00	9.3e-01	1.36e+00	1.0e+00	1.5e+04
13	1.11e+00	9.52e-03	0.00e+00	9.4e-01	1.98e+00	1.0e+00	2.4e+03
14	3.00e-01	1.61e-01	0.00e+00	1.0e+00	1.61e+00	1.0e+00	1.6e+03
15	3.28e-01	3.93e-02	0.00e+00	9.2e-01	1.69e+00	1.0e+00	9.7e+02
16	7.42e-02	7.70e-03	0.00e+00	9.3e-01	1.55e+00	1.0e+00	5.7e+02
17	1.64e-02	3.57e-04	0.00e+00	9.3e-01	1.65e+00	1.0e+00	6.9e+02
18	1.13e-03	7.63e-07	0.00e+00	9.3e-01	1.63e+00	1.0e+00	6.6e+02
19	1.33e-04	1.10e-08	1.11e-16	9.3e-01	1.63e+00	1.0e+00	6.1e+02

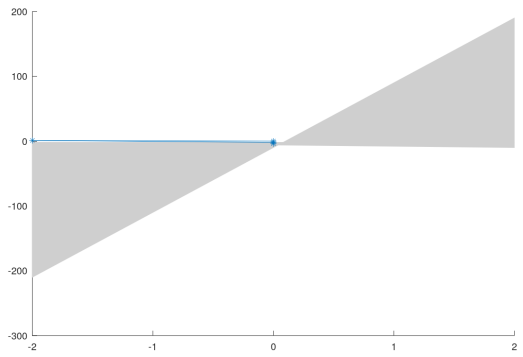
On remarque que la plupart des valeurs de θ sont égales à 1, ce qui confirme notre intuition sur la correction de Powell. Le conditionnement des matrices M_k est d'ordre maximal 10^4, ce qui est relativement important.

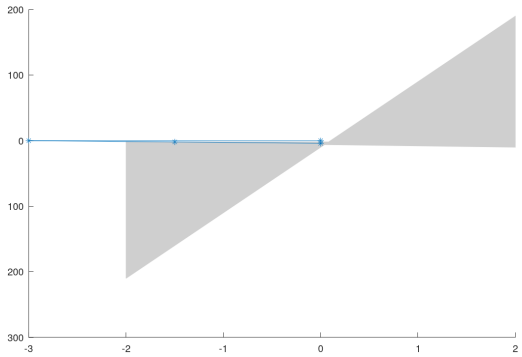
2.4.2 Cas-test 5.b

Le deuxième cas test utilise les données suivantes : on prend 3 barres de longueurs finales `L = [3 2.5 2.5]`. Le deuxième point de fixation de la chaîne est $(a, b) = (0, -4)$, et la position initiale des noeuds est donnée par `xy = [ -2 0 1 -2 ]`.

On décrit un plancher en utilisant deux fonctions linéaires, avec `R = [-6 -10]; S = [-2 100];`

Les graphes suivants représentent la chaîne dans son état initial et final. Cependant, comme le deuxième plancher a un coefficient angulaire très haut, l'échelle verticale des graphes est fortement déformée, alors il est difficile à noter la différence entre la chaîne initiale et finale.





Les **valeurs finales des indicateurs d'optimalité** sont $5.331689e-05$ (gradient) , $8.881784e-16$ (contraintes), et $6.229570e-09$ (minimum entre λ_I et $-c_I$). Le nombre maximal d'itérations autori-

sées est 300 et le **nombre d'itérations effectuées** est 16.

Les **points finaux de la chaîne** sont : `[-3 -1.50009 -6.22957e-09 -2.00007 ]`

Les **multiplicateurs finaux (contraintes d'égalité)** de la chaîne sont `[-6160.56 3359.78 3359.38 ]`

Les **multiplicateurs finaux (contraintes d'inégalité)** de la chaîne sont `[13442.3 0 0 0 ]`.

Voici un tableau avec toute la progression de l'algorithme :

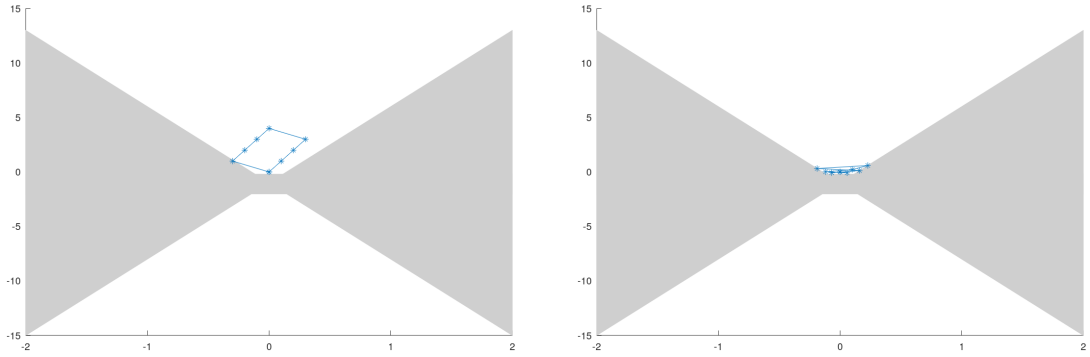
iter	|gl|	|ce|	(ci, lmi)	|x|	|lm|	Powell	cond(M)
1	4.79e+00	2.12e+00	1.11e-16	3.2e+00	1.48e+00	3.4e-01	1.0e+00
2	1.35e+00	2.70e-01	4.44e-16	3.0e+00	7.45e-01	4.3e-01	2.3e+01
3	1.92e+00	2.46e-01	0.00e+00	3.0e+00	1.23e+00	1.1e-01	4.0e+02
4	5.31e-01	5.45e-02	3.54e-16	3.0e+00	7.26e+00	1.0e+00	8.1e+01
5	4.35e-01	1.31e-02	1.84e-16	3.0e+00	1.24e+01	1.0e+00	1.8e+02
6	4.10e-01	3.27e-03	7.81e-18	3.0e+00	2.13e+01	1.0e+00	3.6e+02
7	4.02e-01	8.17e-04	2.17e-19	3.0e+00	3.89e+01	1.0e+00	7.2e+02
8	4.01e-01	2.04e-04	1.08e-19	3.0e+00	7.39e+01	1.0e+00	1.4e+03
9	4.00e-01	5.10e-05	1.36e-20	3.0e+00	1.44e+02	1.0e+00	2.9e+03
10	4.00e-01	1.28e-05	3.39e-21	3.0e+00	2.84e+02	1.0e+00	5.7e+03
11	4.00e-01	3.19e-06	1.69e-21	3.0e+00	5.64e+02	1.0e+00	1.1e+04
12	4.00e-01	7.97e-07	2.12e-22	3.0e+00	1.12e+03	1.0e+00	2.3e+04
13	4.00e-01	1.99e-07	5.29e-23	3.0e+00	2.24e+03	1.0e+00	4.6e+04
14	4.00e-01	4.98e-08	1.32e-23	3.0e+00	4.48e+03	1.0e+00	9.1e+04
15	4.00e-01	1.25e-08	6.62e-24	3.0e+00	8.96e+03	1.0e+00	1.8e+05
16	5.33e-05	8.88e-16	6.23e-09	3.0e+00	1.34e+04	7.5e-02	1.6e+07

On remarque ici aussi que la plupart des valeurs de θ sont égales à 1. Le conditionnement des matrices M_k semble exploser sur les dernières itérations avec un conditionnement d'ordre 10^7 à la dernière itération, ce qui est très important. Cela confirme la complexité du contexte initial et final de l'algorithme.

2.4.3 Cas-test 5.c

Le troisième cas test utilise les données suivantes : on prend 8 barres de longueurs finales `L = [0.1 0.2 0.3 0.4 0.5 0.4 0.3 0.1]`. Le deuxième point de fixation de la chaîne est $(a, b) = (0, 0)$, et c'est à nous de trouver une position initiale des noeuds qui marche. On propose l'utilisation de `xy = [0.1; 0.2; 0.3; 0; -0.1; -0.2; -0.3; 1; 2; 3; 4; 3; 2; 1]`.

On décrit un plancher en utilisant trois fonctions linéaires, avec `R = [-1.0; -0.2; -1.0]; S = [-7.0; 0.0; 7.0];` On remarque que ce plancher a l'allure d'un seau.



Les **valeurs finales des indicateurs d'optimalité** sont $9.569963e-04$ (gradient) , $3.476339e-07$ (contraintes), et $1.110223e-16$ (minimum entre λ_I et $-c_I$). Le nombre maximal d'itérations autorisées est 300 et le **nombre d'itérations effectuées** est 14.

Les **points finaux de la chaîne** sont : `[0.0582848 -0.119134 0.100064 0.227846 -0.188237 0.160164 -0.0693393 -0.0812601 0.0110595 0.215882 0.594923 0.31766 0.121149 -0.0720559 ]`

Les **multiplicateurs finaux (contraintes d'égalité)** de la chaîne sont `[1.47308 -0.484816 0.3913 0.672874 -1.46203 -2.20662 -2.20921 7.31084 ]`

Les **multiplicateurs finaux (contraintes d'inégalité)** de la chaîne sont `[0 0 0 0 0.393457 0 0 0 0 0 0 0 0 0 0 00 0.149295 0 0.364547 0 ]`.

Voici un tableau avec la progression de l'algorithme :

iter	|gl|	|ce|	(ci, lmi)	|x|	|lm|	Powell	cond(M)
1	1.32e+00	6.82e-01	8.88e-16	2.4e+00	1.56e+00	1.0e+00	1.0e+00
2	9.11e-01	4.25e-01	0.00e+00	1.9e+00	1.57e+00	1.0e+00	9.5e+00
3	6.74e-01	1.41e-01	3.05e-16	1.2e+00	5.39e-01	7.0e-01	6.9e+01
4	8.69e-01	5.91e-02	2.22e-16	7.6e-01	3.45e+00	4.6e-01	7.7e+02
5	6.23e-01	1.34e-02	2.50e-16	5.8e-01	3.85e+00	2.4e-01	4.9e+03
6	1.77e+00	3.69e-02	2.22e-16	5.9e-01	9.15e+00	1.0e+00	3.9e+05
7	8.43e-01	3.75e-03	5.55e-17	5.9e-01	7.99e+00	1.0e+00	1.8e+05
8	2.57e-01	1.80e-02	1.11e-16	5.9e-01	8.21e+00	1.0e+00	2.3e+05
9	1.44e-01	2.97e-03	2.22e-16	5.9e-01	7.67e+00	1.0e+00	1.8e+05
10	5.57e-02	5.43e-04	9.71e-17	5.9e-01	6.75e+00	1.0e+00	1.9e+05
11	8.98e-02	2.19e-03	1.11e-16	5.9e-01	6.51e+00	1.0e+00	1.3e+05
12	3.37e-02	9.24e-05	1.11e-16	5.9e-01	7.22e+00	1.0e+00	9.5e+04
13	1.15e-02	5.53e-06	1.80e-16	5.9e-01	7.23e+00	1.0e+00	5.7e+04
14	9.57e-04	3.48e-07	1.11e-16	5.9e-01	7.31e+00	1.0e+00	4.1e+04

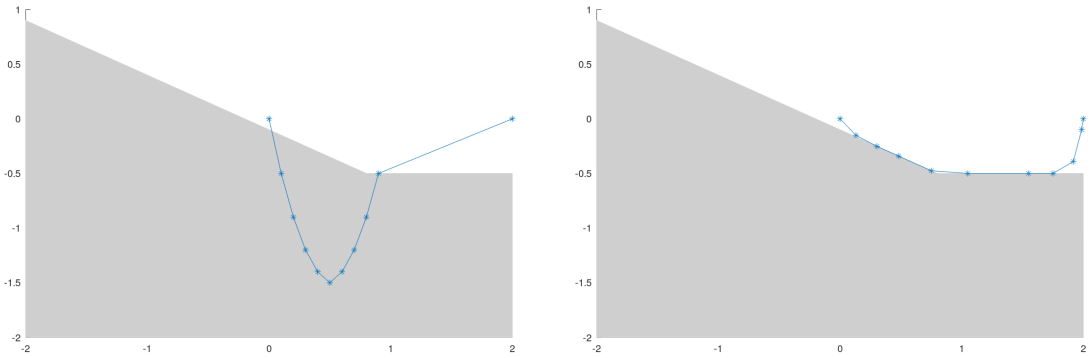
On remarque ici aussi que la plupart des valeurs de θ sont égales à 1. Le conditionnement des matrices M_k est élevé sur de nombreuses itérations : le conditionnement est d'ordre 10^4 à 10^5 dans 8 itérations sur 14. C'est relativement important et cela confirme là aussi la complexité du contexte initial et final de l'algorithme.

2.4.4 Cas-test 5.d

On remarque que l'algorithme développé ici fonctionne bien dans les cas-test précédents, et l'objectif de ce cas-test et le suivant est de faire cette vérification.

C'est-à-dire, on reprend un cas-test déjà étudié précédemment dans la partie 1.5.4 : on prend 10 barres de longueur final `L = [0.2; 0.2; 0.2; 0.3; 0.3; 0.5; 0.2; 0.2; 0.3; 0.1]`. Le deuxième point de fixation de la chaîne est $(a, b) = (2, 0)$, et la position initiale des noeuds est donnée par `xy = [0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; −0.5; −0.9; −1.2; −1.4; −1.5; −1.4; −1.2; −0.9; −0.5]`.

On décrit un plancher en utilisant deux fonctions linéaires, avec `R = [−0.1; −0.5]; S = [−0.5; 0];`



Les **valeurs finales des indicateurs d'optimalité** sont $1.818879e-04$ (gradient) , $2.272237e-07$ (contraintes), et $1.110223e-16$ (minimum entre λ_I et $-c_I$). Le nombre maximal d'itérations autorisées est 300 et le **nombre d'itérations effectuées** est 21.

Les **points finaux de la chaîne** sont : `[0.130743 0.303667 0.482552 0.75088 1.04987 1.54987 1.74987 1.91763 1.98616 −0.151349 −0.251833 −0.341276 −0.47544 −0.5 −0.5 −0.5 −0.391107 −0.0990383 ]`

Les **multiplicateurs finaux (contraintes d'égalité)** de la chaîne sont `[1.32511 1.00159 0.77593 0.331018 0.115662 0.0691736 0.172921 0.206342 0.505031 2.49906 ]`

Les **multiplicateurs finaux (contraintes d'inégalité)** de la chaîne sont `[0 0.137479 0.200038 0.216907 0 0 0 0 00 0 0 0 0.3943 0.349972 0.15505 0 0 ]`.

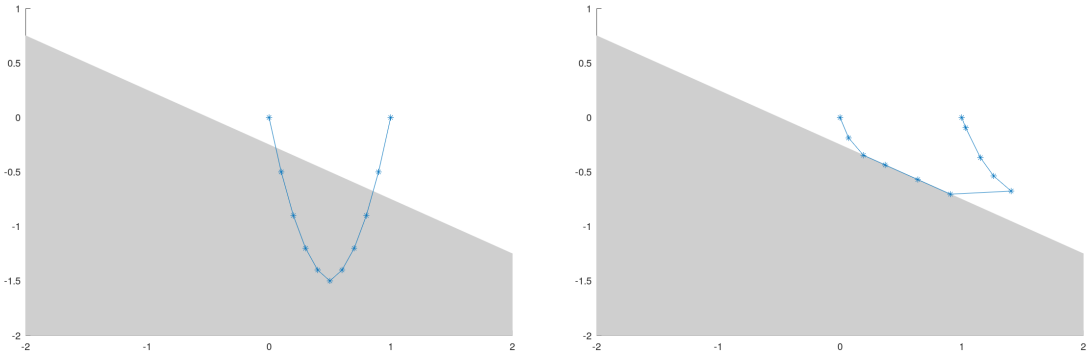
Voici un tableau avec la progression de l'algorithme :

iter	|gl|	|ce|	(ci, lmi)	|x|	|lm|	Powell	cond(M)
1	1.27e+02	3.76e+00	4.44e-16	3.7e+00	5.13e+01	8.0e-02	1.0e+00
2	4.87e+01	1.05e+00	4.44e-16	2.6e+00	6.81e+01	1.0e+00	2.7e+01
3	2.55e+01	4.55e-01	3.33e-16	2.2e+00	3.73e+01	1.0e+00	4.1e+01
4	7.11e+00	3.01e-01	0.00e+00	2.1e+00	9.90e+00	1.0e+00	3.7e+01
5	2.48e+00	1.02e-01	0.00e+00	1.9e+00	7.70e+00	1.0e+00	4.3e+01
6	1.23e+00	2.17e-02	0.00e+00	2.0e+00	5.67e+00	1.0e+00	3.9e+01
7	5.68e-01	9.41e-03	0.00e+00	2.0e+00	4.05e+00	9.8e-01	1.1e+02
8	4.13e-01	1.75e-03	0.00e+00	2.0e+00	6.50e-01	7.9e-01	1.8e+02
9	6.42e-01	1.69e-02	0.00e+00	2.1e+00	3.14e+00	6.5e-01	9.3e+02
10	1.43e+00	4.69e-02	0.00e+00	2.1e+00	4.98e+00	1.0e+00	2.1e+02
11	1.03e+00	1.39e-02	5.55e-17	2.1e+00	3.83e+00	1.0e+00	1.5e+02
12	9.26e-01	3.38e-03	2.78e-17	2.1e+00	4.69e+00	9.8e-01	2.7e+02
13	4.63e-01	1.97e-03	5.55e-17	2.1e+00	8.84e-01	8.6e-01	3.4e+02
14	3.97e-01	2.62e-03	5.55e-17	2.1e+00	1.59e+00	5.5e-01	5.7e+02
15	4.61e-01	5.72e-03	2.78e-17	2.1e+00	2.34e+00	3.9e-01	2.8e+03
16	8.05e-01	3.13e-03	5.55e-17	2.0e+00	2.43e+00	1.0e+00	8.6e+02
17	1.65e-01	1.66e-03	5.55e-17	2.0e+00	3.41e+00	1.0e+00	6.8e+02
18	1.18e-01	1.61e-04	0.00e+00	2.0e+00	3.14e+00	1.0e+00	7.4e+02
19	1.44e-02	1.11e-05	5.55e-17	2.0e+00	2.54e+00	1.0e+00	7.1e+02
20	2.34e-03	5.69e-06	5.55e-17	2.0e+00	2.49e+00	1.0e+00	7.1e+02
21	1.82e-04	2.27e-07	1.11e-16	2.0e+00	2.50e+00	1.0e+00	7.1e+02

La plupart des valeurs de θ sont égales à 1 et le conditionnement des matrices M_k reste raisonnable au fil des itérations avec un ordre maximal de 10^2. On peut donc considérer ce problème comme bien posé.

2.4.5 Cas-test 5.e (extra)

On reprend le cas-tests 4b déjà étudié ici. On prend exactement les mêmes données précédents, en changeant seulement la description du plancher : `R = -0.25; S = -0.5`.



Les **valeurs finales des indicateurs d'optimalité** sont $3.666398e-04$ (gradient) , $1.717822e-07$ (contraintes), et $3.195225e-09$ (minimum entre λ_I et $-c_I$). Le nombre maximal d'itérations autorisées est 300 et le **nombre d'itérations effectuées** est 16.

Les **points finaux de la chaîne** sont : `[0.0702593 0.191947 0.370832 0.63916 0.907489 1.40669 1.26263 1.15388 1.03308 -0.187253 -0.345973 -0.435416 -0.56958 -0.703744 -0.675553 -0.536825 -0.368973 -0.0943714 ]`

Les **multiplicateurs finaux (contraintes d'égalité)** de la chaîne sont `[1.04268 0.600511 0.316525 0.0244629 -0.199353 -0.344198 1.19191 1.57997 1.42119 5.1946 ]`

Les **multiplicateurs finaux (contraintes d'inégalité)** de la chaîne sont `[0 0.0662887 0.200043 0.240053 0.4730780 0 0 0 ]`.

Voici un tableau avec la progression de l'algorithme :

iter	|gl|	|ce|	(ci, lmi)	|x|	|lm|	Powell	cond(M)
1	2.90e+02	8.86e+00	4.44e-16	4.1e+00	3.80e+01	1.7e-02	1.0e+00
2	2.39e+01	2.16e+00	6.66e-16	2.8e+00	1.29e+01	1.0e+00	2.7e+01
3	5.16e+00	4.86e-01	2.00e-15	2.1e+00	6.96e+00	1.0e+00	2.5e+01
4	2.36e+00	1.22e-01	9.99e-16	1.8e+00	6.95e+00	1.0e+00	2.7e+01
5	1.15e+00	3.89e-02	1.11e-16	1.6e+00	1.02e+01	1.0e+00	4.1e+01
6	5.42e-01	8.58e-03	0.00e+00	1.5e+00	6.02e+00	1.0e+00	7.2e+01
7	2.61e-01	1.38e-03	0.00e+00	1.4e+00	2.27e+00	1.0e+00	9.5e+01
8	1.53e-01	4.32e-03	5.55e-17	1.4e+00	5.09e+00	1.0e+00	9.2e+01
9	1.35e-01	1.51e-03	5.55e-17	1.4e+00	5.76e+00	1.0e+00	1.0e+02
10	1.11e-01	2.42e-04	5.55e-17	1.4e+00	5.40e+00	9.5e-01	9.7e+01
11	9.01e-02	2.17e-03	1.11e-16	1.4e+00	4.90e+00	1.0e+00	1.0e+02
12	5.61e-02	1.45e-04	5.55e-17	1.4e+00	5.08e+00	1.0e+00	1.5e+02
13	1.28e-02	2.95e-05	5.55e-17	1.4e+00	5.18e+00	1.0e+00	1.5e+02
14	2.58e-03	1.39e-06	1.11e-16	1.4e+00	5.20e+00	1.0e+00	1.5e+02
15	1.71e-03	1.90e-07	1.11e-16	1.4e+00	5.20e+00	1.0e+00	1.5e+02
16	3.67e-04	1.72e-07	3.20e-09	1.4e+00	5.19e+00	1.0e+00	1.5e+02

De même que dans le cas-test précédent, on remarque que la plupart des valeurs de θ sont égales à 1 et que le conditionnement des matrices M_k reste raisonnable au fil des itérations, avec un ordre maximal de 10^2. On peut donc ici aussi considérer ce problème comme bien posé.

Enfin, on observe dans ces deux derniers cas-tests que le nouvel algorithme donne les mêmes chaînes finales que précédemment.

Conclusion

Dans ce projet, on a étudié une introduction pratique à l'optimisation quadratique successive, en implémentant un algorithme qui permet de déduire la position d'équilibre d'une chaîne suspendue au dessus d'un plancher convexe. On a utilisé la factorisation de Cholesky et la formule de BFGS pour calculer la matrice approchant le hessien du lagrangien.

Une possible direction d'exploration est la globalisation de cet algorithme ou bien une autre manière de substituer la matrice hessienne dans le problème quadratique osculateur. Par exemple, au lieu d'utiliser la factorisation de Cholesky modifiée ou la formule de BFGS, on pourrait prendre la formule *SR1* (Symétrique de Rang 1).