Équilibre d'une chaîne articulée - TP4

Aristote Koen
Thomas Chatrefou

Février 2020

Introduction

On s'intéresse au problème de trouver la position d'équilibre d'une chaîne formée de barres rigides fixées à ses deux bouts. Cette position correspond au minimum de l'énergie potentielle sous certaines contraintes.

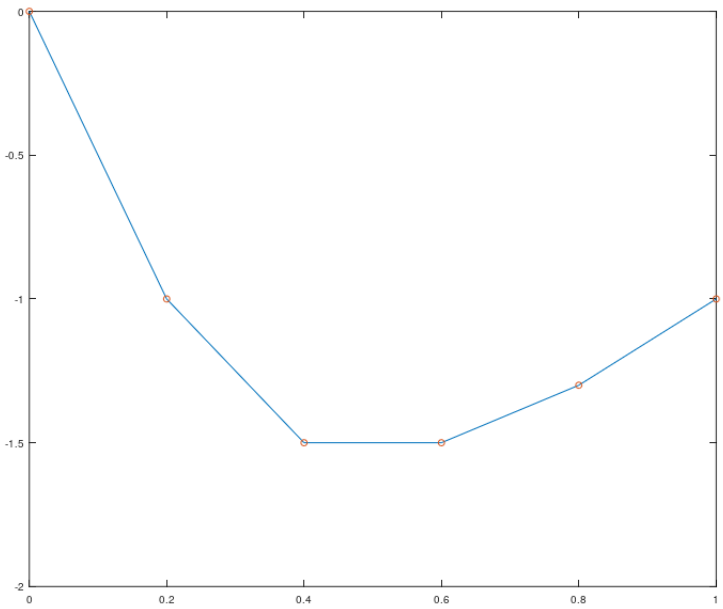


FIGURE 1 – Chaîne formée de barres

On suppose que les extrémités de la chaîne sont fixées aux points $(0, 0)$ et (a, b) où on pourra faire varier a et b sur plusieurs exemples. On note n_b le nombre de barres et L_i la longueur de la i-ème barre, $n_n = n_b - 1$ le nombre de nœuds non fixés et (x_i, y_i) les coordonnées du i-ème nœuds.
L'objectif est donc de minimiser la fonction Énergie potentielle de la forme suivante :

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

où

$$l_i(x, y) = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

avec la convention $(x_0, y_0) = (0, 0)$ et $(x_{n_b}, x_{n_b}) = (a, b)$.
Les contraintes à respecter portent sur la longueur des barres et s'écrivent donc :

$$\forall i \in [\![1, n_b]\!], \quad c_i(x, y) = l_i^2(x, y) - L_i^2 = 0 \tag{2}$$

Maintenant, comme $l_i = L_i$ sur l'ensemble admissible, on peut résoudre plus simplement le problème linéaire suivant :

$$(P) \quad \begin{cases} \min\ e(x, y) := \dfrac{1}{2} \displaystyle\sum_{i=1}^{n_b} L_i(y_i - y_{i-1}) \\ c_i(x, y) = 0, \quad \forall i \in [\![1, n_b]\!] \end{cases}$$

Clairement, pour que le problème soit bien définit il faut que la longueur de la chaîne soit plus grande que la distance séparant les deux extrémités c'est à dire qu'on ne peut pas avoir $\sum_{i=1}^{n_b} L_i < (A^2 + B^2)^{1/2}$. De plus la fonction coût étant continue $\forall (x, y) \in \mathbb{R}^{2n_n}$ elle est donc semi-continue inferieurement. De plus $\forall L_i$ t.q $\sum_{i=1}^{n_b} L_i \geq (A^2 + B^2)^{1/2}$ l'ensemble admissible :

$$X = \{(x, y) \in \mathbb{R}^{n_n} \times \mathbb{R}^{n_n} \ : \ c_i(x, y) = 0 \ \forall i \in [\![1, n_b]\!]\}$$

est fermé car c'est l'image réciproque d'un singleton par une fonction continue. Cet ensemble étant borné il est donc compact. Ainsi il existe au moins une solution au problème (P).

On écrira par la suite $E = [\![1, n_b]\!]$, et $\xi = (x^T\, y^T)^T$ la concaténation du vecteur abscisses $(x_i)_{i \in E}$ et du vecteur ordonnée $(y_i)_{i \in E}$.
On note alors le Lagrangien du problème :

$$l(\xi, \lambda) = e(\xi) + \lambda^T c(\xi) \tag{3}$$

où $\lambda \in \mathbb{R}^{n_b}$.

On peut observer dans la section "Jacobienne des contraintes" ci dessous que la contrainte est une fonction affine donc surjective. De plus, par les conditions nécessaires d'optimalité du premier ordre avec contraintes d'égalité on sait que si x_*est une solution du problème et que le "gradient" de la contraintes est surjectif alors $\exists \lambda_* \in \mathbb{R}^{n_b} \ : \ \nabla l(x_*, \lambda_*) = 0$

Simulateur

Afin de créer notre simulateur nous avons écrit une fonction sous `Matlab` permettant de tracer la chaîne et de calculer la valeur de l'énergie potentielle, la valeur des contraintes sur la longueur des barres, le gradient de l'énergie, la Jacobienne des contraintes d'égalité ainsi que le Hessien du Lagrangien.

Structure générale du simulateur

La fonction `Matlab` se présente comme suit :

```
function [e, c, g, a, hl, indic] = chs(indic, xy, lm)
```

Elle prend en entrée :

- `xy` : le vecteur ξ comportant les abscisses et les ordonnées des nœuds internes ;
- `indic` : qui est un entier pilotant le comportement du simulateur ;
- `lm` : le vecteur λ contenant les multiplicateurs de Lagrange ;

et retourne :

- `e` : la valeur de l'énergie potentielle en `xy` ;
- `c` : la valeur en `xy` des contraintes sur la longueur des barres, c'est un vecteur colonne de taille n_b ;
- `g` : le gradient de `e` en `xy` ;
- `a` : la Jacobienne des contraintes d'égalité en `xy`, une matrice de taille $(n_b, 2n_n)$;
- `hl` : le Hessien du Lagrangien en `xy`, une matrice de taille $2n_n$;
- `indic` : décrivant ici le résultat de la simulation, il vaut :
  1 si les paramètres d'entrée sont incorrects
  0 si la sortie est normale

Tracé de la chaîne articulée

Si `indic` $= 2$ en entrée alors la fonction fera un tracé de la chaîne, comme sur la Figure 1, à partir du vecteur `xy` et des extrémités $(0, 0)$ et $(\mathtt{A}, \mathtt{B})$ où `A` et `B`, déclarées dans le fichier `ch.m`, correspondent aux paramètres (a, b).

Énergie potentielle et contraintes sur la longueur des barres

Dans le cas où `indic` $= 3$ la fonction retournera `e` et `c` qui sont calculées de manière vectorielle :

$$\mathtt{e} = \frac{1}{2} L \cdot \left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_n} \\ b \end{bmatrix} - \begin{bmatrix} 0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n_n} \end{bmatrix} \right)$$

et le vecteur des contraintes :

$$\mathtt{c} = \begin{bmatrix} (x_1 - 0)^2 \\ (x_2 - x_1)^2 \\ \vdots \\ (a - x_{n_n})^2 \end{bmatrix} + \begin{bmatrix} (y_1 - 0)^2 \\ (y_2 - y_1)^2 \\ \vdots \\ (b - y_{n_n})^2 \end{bmatrix} - \begin{bmatrix} L_1^2 \\ L_2^2 \\ \vdots \\ L_{n_b}^2 \end{bmatrix}$$

Gradient de l'énergie potentielle et Jacobienne des contraintes

Si `indic` $= 4$, en plus de `e` et `c`, la fonction calcule `g` et `a`.
`g` étant le gradient de `e`, son expression est simplement :

$$\mathtt{g} = \frac{1}{2} (\underbrace{0 \ , \ \dots \ , \ 0}_{n_n \text{termes}} \ , \ L_1 - L_2 \ , \ \dots \ , \ L_{n_n} - L_{n_b} \ )^T$$

Ensuite, les termes de la Jacobienne des contraintes sont donnés par dérivation de (2) :

$$\forall (i, j) \in [\![1, n_n]\!] \times E, \quad \partial_j c_i = 2(\xi_i - \xi_{i-1})(\delta_{ij} - \delta_{i-1,\,j}) + 2(\xi_{n_n+i} - \xi_{n_n+i-1})(\delta_{n_n+i,\,j} - \delta_{n_n+i-1,\,j})$$

où δ_{ij} est le symbole de Kronecker valant 1 si $i = j$ et 0 si non. On a donc la matrice $\mathtt{a} \in \mathcal{M}_{n_b,\,2n_n}(\mathbb{R})$:

$$\mathtt{a} = 2\begin{pmatrix} x_1 & & & & & y_1 & & & \\ -x_1 & x_2 - x_1 & & & & -y_1 & y_2 - y_1 & & \\ & -x_2 + x_1 & \ddots & & & & -y_2 + y_1 & \ddots & \\ & & \ddots & x_{n_n} - a & & & & \ddots & y_{n_n} - b \\ & & & -x_{n_n} + a & & & & & -y_{n_n} + b \end{pmatrix}$$

Hessien du Lagrangien

Dans le cas où `indic` = 5, le simulateur retournera le Hessien du Lagrangien associé aux contraintes d'égalité en `lm` et `xy` :

$$\nabla^2 l = \nabla^2 e + \sum_{i=1}^{n_b} \lambda_i \nabla^2 c_i$$

L'énergie potentielle étant définie comme une fonction affine, son hessien est nul et donc il suffit de calculer :

$$\nabla^2 l = \sum_{i=1}^{n_b} \lambda_i \nabla^2 c_i$$

où pour $i \in E$ fixé on a : $\forall i, j \in [\![1, n_n]\!]$,

$$\begin{aligned}(\nabla^2 c_i)_{jk} &= \partial_k \partial_j c_i \\ &= 2(\delta_{ij} - \delta_{i-1,\,j})(\delta_{ik} - \delta_{i-1,\,k}) + 2(\delta_{n_n+i,\,j} - \delta_{n_n+i-1,\,j})(\delta_{n_n+i,\,k} - \delta_{n_n+i-1,\,k}) \\ &= 2\delta_{ij}(\delta_{jk} - \delta_{j-1,\,k}) - 2\delta_{i-1,\,j}(\delta_{j+1,\,k} - \delta_{jk}) + 2\delta_{n_n+i,\,j}(\delta_{jk} - \delta_{j-1,\,k}) - 2\delta_{n_n+i-1,\,j}(\delta_{j+1,\,k} - \delta_{jk})\end{aligned}$$

Donc les matrices Hessiennes associées aux contraintes c_i ont la forme suivante :

$$\nabla^2 c_1 = \begin{pmatrix} M_1 & 0 \\ 0 & M_1 \end{pmatrix} \quad \text{avec } M_1 \in \mathcal{M}_{n_n}(\mathbb{R}) \text{ tq } M_1 = \begin{pmatrix} 2 & 0 & \dots & 0 \\ 0 & 0 & & \\ \vdots & & \ddots & \vdots \\ 0 & & \dots & 0 \end{pmatrix}$$

$$\nabla^2 c_2 = \begin{pmatrix} M_2 & 0 \\ 0 & M_2 \end{pmatrix} \quad \text{avec } M_2 \in \mathcal{M}_{n_n}(\mathbb{R}) \text{ tq } M_2 = \begin{pmatrix} 2 & -2 & 0 & \dots & 0 \\ -2 & 2 & 0 & & \\ 0 & 0 & 0 & & \\ \vdots & & & \ddots & \vdots \\ 0 & & & \dots & 0 \end{pmatrix}$$

$$\nabla^2 c_i = \begin{pmatrix} M_i & 0 \\ 0 & M_i \end{pmatrix} \quad \text{avec } M_i \in \mathcal{M}_{n_n}(\mathbb{R}) \text{ tq } M_i = \begin{pmatrix} 0 & \dots & & & & 0 \\ \vdots & \ddots & & & & \\ & & 2 & -2 & & \\ & & -2 & 2 & & \\ & & & & \ddots & \vdots \\ 0 & & & & \dots & 0 \end{pmatrix}$$

$$\nabla^2 c_{n_b} = \begin{pmatrix} M_{n_b} & 0 \\ 0 & M_{n_b} \end{pmatrix} \quad \text{avec } M_{n_b} \in \mathcal{M}_{n_n}(\mathbb{R}) \text{ tq } M_{n_b} = \begin{pmatrix} 0 & \dots & & 0 \\ \vdots & \ddots & & \vdots \\ & & 0 & 0 \\ 0 & \dots & 0 & 2 \end{pmatrix}$$

Ainsi nous obtenons la matrice `hl` calculée par le simulateur :

$$\nabla^2 l = \sum_{i \in E} \lambda_i \nabla^2 c_i = 2 \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix}$$

où $M \in \mathcal{M}_{n_n}(\mathbb{R})$ s'écrit

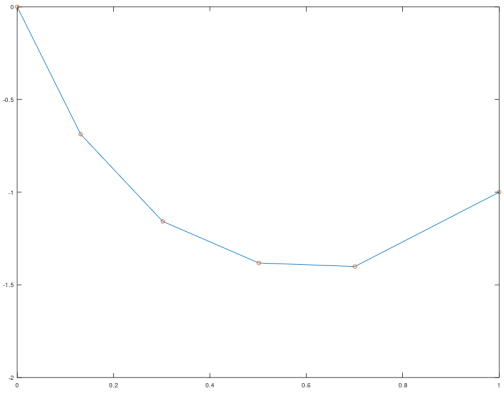
$$M = \begin{pmatrix} \lambda_1 + \lambda_2 & -\lambda_2 & & & \\ -\lambda_2 & \lambda_2 + \lambda_3 & -\lambda_3 & & \\ & -\lambda_3 & \lambda_3 + \lambda_4 & \ddots & \\ & & \ddots & \ddots & -\lambda_{n_n} \\ & & & -\lambda_{n_n} & \lambda_{n_n} + \lambda_{n_b} \end{pmatrix}$$

Optimiseur

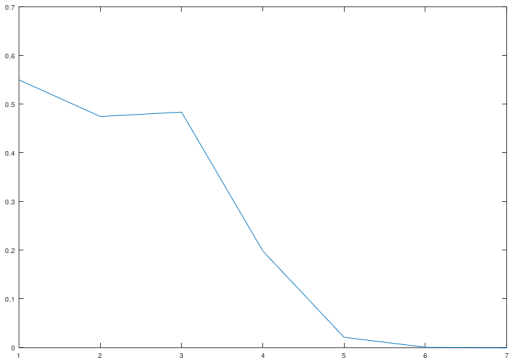
On a testé l'optimiseur sur les quatres cas-tests assez similaires : avec 5 barres de longueur respectives $0.7, 0.5, 0.3, 0.2$ et 0.7, et de points de fixation $(0, 0)$ et $(1, -1)$ mais avec positions initiales différentes :

— **cas-test 2a** sur la Figure : 2 $(0.2, -1)$, $(0.4, -1.5)$, $(0.6, -1.5)$, et $(0.8, -1.3)$
— **cas-test 2b** sur la Figure : 3 $(0.2, 1)$, $(0.4, 1.5)$, $(0.6, 1.5)$, et $(0.8, 1.3)$
— **cas-test 2c** sur la Figure : 4 $(0.2, -1)$, $(0.4, -1.5)$, $(0.6, 1.5)$, et $(0.8, -1.3)$
— **cas-test 2d** sur la Figure : 5 $(0.2, 1)$, $(0.4, -1.2)$, $(0.6, 1.5)$, et $(0.8, -1.3)$

Les résultats sont représentés sur les figures ci-dessous, avec la chaîne articulée à gauche et une représentation de la convergence de la méthode au cours des itérations.



(a) Chaîne articulée

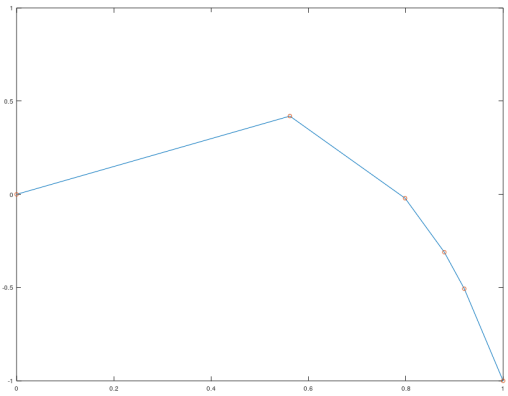


(b) $\|F(z_{k+1}) - F(z_k)\|_\infty$ en fonction de l'itération k

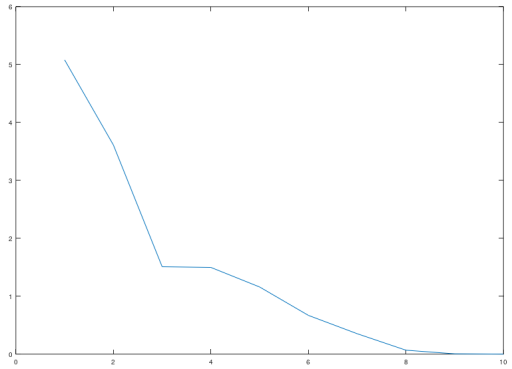
FIGURE 2 – Cas-test 2a

Dans le cas 2a l'algorithme converge vers un minimum. En effet on observe sur la Figure 2 que la chaîne est dans un état minimisant l'énergie potentielle. Les données du tableau ci-dessus confirment bien la convergence, on atteint nos niveaux de tolérance sur les deux critères en x itérations.

Dans le cas 2b l'algorithme converge vers un maximum. En effet on observe sur la Figure 3 que



(a) Chaîne articulée



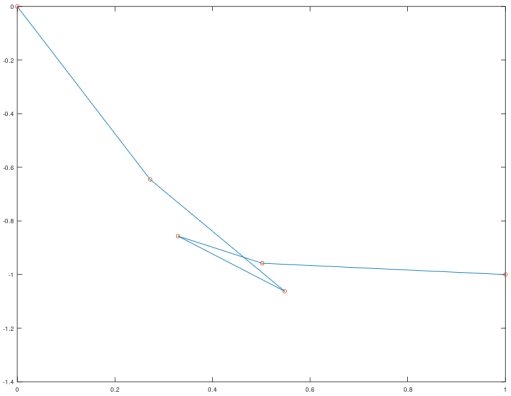
(b) $\|F(z_{k+1}) - F(z_k)\|_\infty$ en fonction de l'itération k

FIGURE 3 – Cas-test 2b

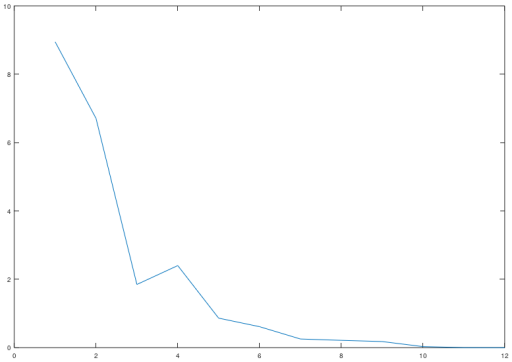
la chaîne est dans un état maximisant l'énergie potentielle. Les données du tableau ci-dessus

confirment bien la convergence, on atteint nos niveaux de tolérance sur les deux critères en x itérations.

Dans le cas 2c l'algorithme semble converger vers un point selle caractérisé par la boucle formée



(a) Chaîne articulée

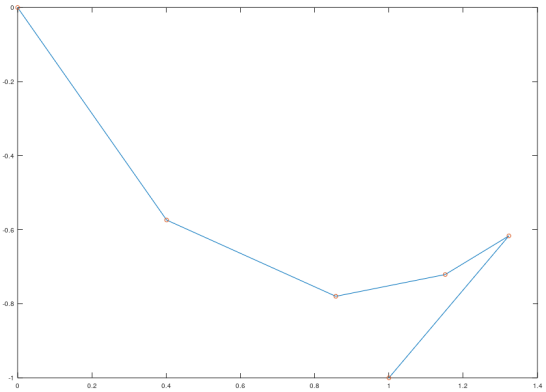


(b) $\|F(z_{k+1}) - F(z_k)\|_\infty$ en fonction de l'itération k

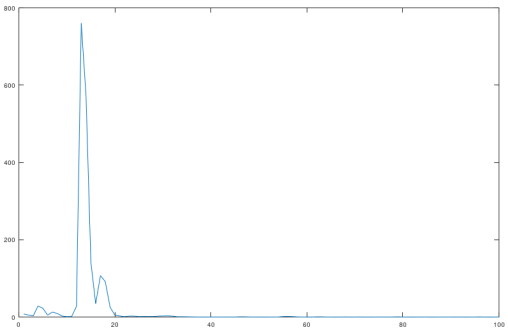
FIGURE 4 – Cas-test 3c

sur la chaîne que l'on peut observer sur la Figure 4.

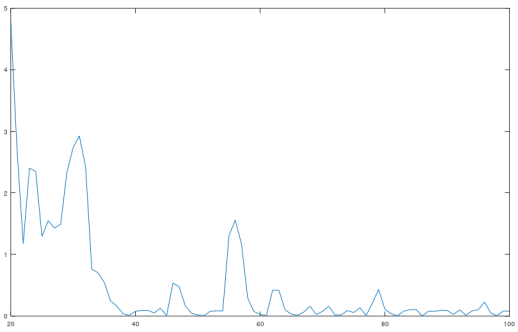
Dans le cas 2d et pour lm = [0 0 0 0 0] l'algorithme ne converge pas. En effet on observe bien



(a) Chaîne articulée



(b) $\|F(z_{k+1}) - F(z_k)\|_\infty$ en fonction de l'itération k



(c) $\|F(z_{k+1}) - F(z_k)\|_\infty$ entre 20 et 100 itérations

FIGURE 5 – Cas-test 2d

sur le graphe c de la Figure 5 que $\|F(x_k) - F(x_{k+1})\|_\infty$ au cours des itérations oscille en dehors

de notre seuil de tolérance et après plusieurs tests pour un nombre plus grand d'itérations nous n'observons pas de convergence. En choisissant lm =[1 1 1 1 1] dans ce même cas et pour ce même algorithme, on peut observer une convergence vers un maximum sur la figure 6b après 44 itérations.

	`tol`	2a	2b	2c	2d
$\|\nabla_x l(x_f, \lambda_f)\|_\infty$	10^{-7}	4.10^{-12}	4.10^{-10}	1.10^{-11}	4.10^{-2}
$\|c(x_f)\|_\infty$	10^{-7}	0	0	0	0
nb itérations	100	7	10	12	100

x_f est le point atteint à la dernière itération.

Dans le cas 2d l'algorithme ne converge pas. En effet on observe bien sur le graphe (c) de la Figure 5 que $\|F(x_k) - F(x_{k+1})\|_\infty$ au cours des itérations oscille en dehors de notre seuil de tolérance et après plusieurs tests pour un nombre plus grand d'itérations nous n'observons pas de convergence.

Convergence de la méthode

Nous pouvons évaluer la convergence de z_k vers sa limite z_* en examinant le comportement de $\|F(z_k)\|_\infty = \max(\|\nabla_x l(x_k, \lambda_k)\|_\infty, \|c(x_k)\|_\infty)$. En effet la solution du problème z_* est un point stationnaire et $F(z_*) = 0$ donc il suffit d'étudier $\|F(z_k) - F(z_{k+1})\|_\infty$ au cours des itérations afin d'estimer sa vitesse convergence vers 0 comme nous l'avons fait sur les tests précédents afin d'observer cette vitesse de convergence.

L'utilité d'observer cette norme au lieu de $\|z_k - z_*\|$ est que nous n'avons pas besoin de connaître la solution exacte z_* afin d'étudier la vitesse de convergence.

Globalisation par recherche linéaire

La direction $p_k = (d_k, \mu_k)$ est une direction de descente en $z_k = (x_k, \lambda_k)$ de la fonction $\phi : \mathbb{R}^{n+m} \to \mathbb{R}$ définie en $z = (x, \lambda)$ par

$$\phi(z) = \frac{1}{2} \|F(z)\|^2$$

En effet,

$$\phi'(z_k).p_k = (F(z_k) \cdot \nabla F(z_k)) \cdot p_k$$

Or : $\nabla F(z_k) \cdot p_k = -F(z_k)$ ainsi :

$$\phi'(z_k) \cdot p_k = -\|F(z_k)\|^2 < 0 \text{ (si } F(z_k) = 0 \text{ alors } z_k = z_*)$$

Au lieu de mettre à jour z_k par le système linéaire intervenant dans le TP2, on le fait par $x_{k+1} := x_k + \alpha_k d_k$ et $\lambda_{k+1} := \lambda_k + \alpha_k \mu_k$ où $p_k = (d_k, \mu_k)$ est toujours déterminé par le système linéaire du TP2 et $\alpha_k > 0$ est un pas déterminé par recherche linéaire sur ϕ le long de p_k. On utilise la règle de recherche d'Armijo : on prend α_k de la forme 2^{-i_k} , où i_k est le plus petit entier positif tel que l'on ait :

$$\phi(z_k + \alpha_k p_k) \leq \phi(z_k) + \omega \alpha_k \phi'(z_k) \cdot p_k \text{ où } \omega \text{ est pris dans }]0, 1, 2[\text{ (typiquement } \omega = 10^{-4})$$

Comme p_k est une direction de descente, on peut commencer la recherche avec $\alpha_k = 1$ en effet, si l'inégalité n'est pas vérifiée, comme la pente $\phi'(z_k) \cdot p_k$ est négative on peut toujours augmenter i_k de sorte à diminuer α_k et s'assurer que l'inégalité soit vérifiée.

En effet, montrons que l'on peut toujours trouver $\alpha_k > 0$ telle que l'inégalité soit vérifiée. Raisonnons par l'absurde et supposons donc que il existe une suite $\{\alpha_{k_i}\}_{i\geq 1}$ telle que $\alpha_{k_i} \to 0$ $i \to \infty$ et telle que $\phi(z_k + \alpha_{k_i} p_k) > \phi(z_k) + \omega \alpha_{k_i} \phi'(z_k) \cdot p_k$ alors comme $\alpha_{k_i} > 0$ $\forall i \geq 1$ on a :

$$\frac{\phi(z_k + \alpha_{k_i} p_k) - \phi(z_k)}{\alpha_{k_i}} > \omega \phi'(z_k) p_k > \phi'(z_k) p_k \quad (0 < \omega < \frac{1}{2})$$

et en passant à la limite $i \to \infty$ càd $\alpha_{k_i} \to 0$ on a donc :

$$\phi'(z_k).p_k > \phi'(z_k).p_k$$

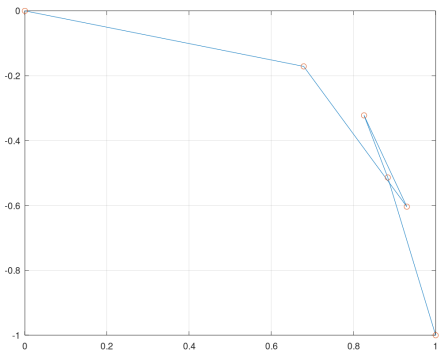
ce qui est absurde. Ainsi on peut toujours trouver un $\alpha_k > 0$ de sorte que l'inégalité du critère d'Armijo soit vérifiée.

Résultats de l'optimiseur pour la globalisation par recherche linéaire

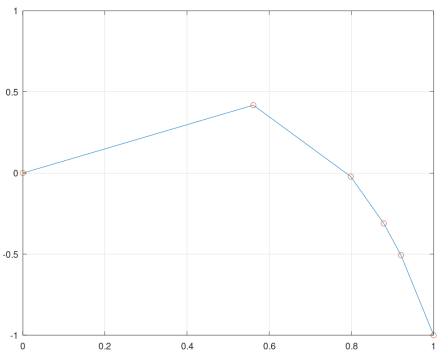
On a testé l'optimiseur sur les quatres cas-tests suivants :

- **cas-test 2d** sur la Figure : 5 comme dans le cas précédent
- **cas-test 3a** sur la Figure : 2 2 barres de longueur 0.6 et 0.6. Deuxième point de fixation de la chaîne $(1, 0)$. Position initiale du noeud $(0.5, 0.4)$
- **cas-test 3b** sur la Figure : 3 3 barres de longueur 0.5, 2 et 0.5. Deuxième point de fixation de la chaîne $(1, 0)$. Position initiale du noeud $(0.3, -0.2)$, $(0.7, -0.2)$
- **cas-test 3c** sur la Figure : 4 3 barres de longueur 0.5, 2 et 0.5. Deuxième point de fixation de la chaîne $(0, -1)$. Position initiale du noeud $(0.5, 0)$, $(0.5, -1)$

Les résultats sont représentés sur les figures ci-dessous par la chaîne articulée à la fin des itérations.



(a) Cas 2d avec recherche linéaire



(b) cas 2d avec recherche par pas unité

FIGURE 6 – Cas-test 2d

On observe dans le cas 2d à recherche linéaire que l'algorithme converge vers un point selle que l'on peut voir sur la Figure 6a après 9 itérations (on peut vérifier en faisant tourner l'optimiseur, options.verb =1 ou 2, que les contraintes sont bien satisfaites et que la norme du gradient du

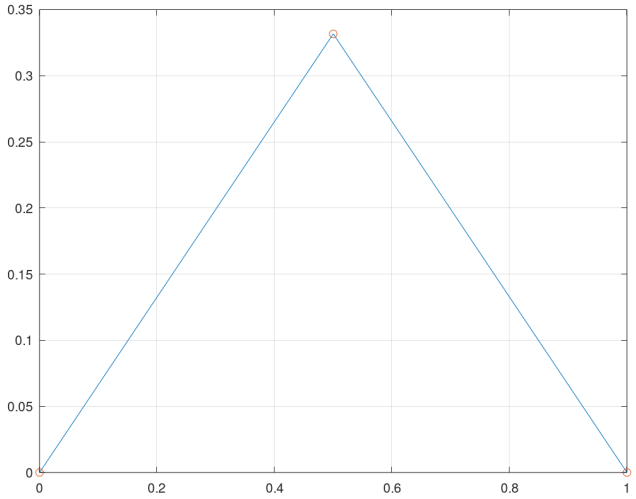


FIGURE 7 – Cas-test 3a avec recherche linéaire

Lagrangien s'annule) alors que dans le cas à pas unité on converge vers un maximum en 44 itérations (cf Figure 6b).

Dans le cas 3a, l'algorithme converge vers un maximum que l'on peut voir sur la Figure 7 en seulement trois itérations, en effet le point de départ de la chaîne articulée est proche de cet état qui est un zéro de la fonction F. De plus on peut vérifier en faisant tourner le simulateur (options.verb= 1 ou 2) que la norme du gradient du Lagrangien s'annule et que les contraintes sont respectées. Dans le cas 3b, l'algorithme converge vers un minimum que l'on peut voir sur la

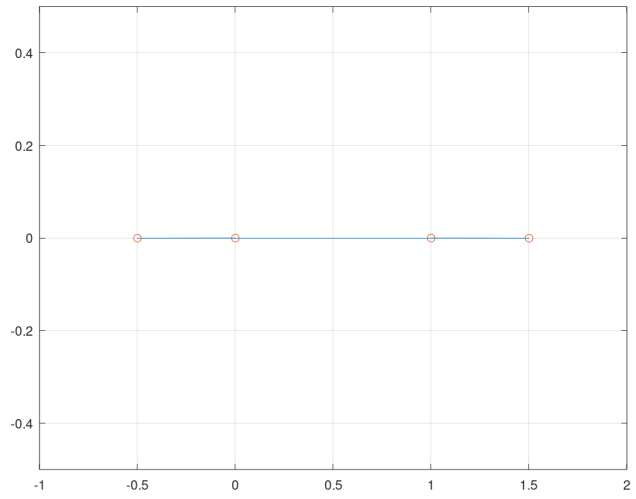


FIGURE 8 – Cas-test 3b avec recherche linéaire

Figure 8 en 33 itérations. De plus on peut vérifier en faisant tourner le simulateur (options.verb= 1 ou 2) que la norme du gradient du Lagrangien s'annule et que les contraintes sont respectées.

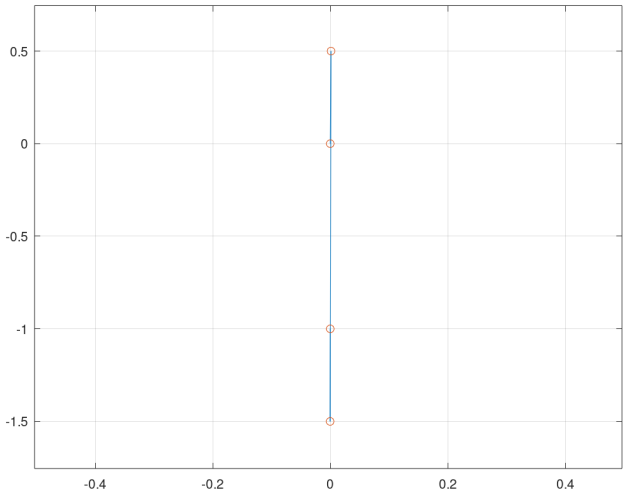


FIGURE 9 – Cas-test 3c avec recherche linéaire

Dans le cas 3c, l'algorithme converge vers un minimum que l'on peut voir sur la Figure 9 en 16 itérations. De plus on peut vérifier en faisant tourner le simulateur (`options.verb` = 1 ou 2) que la norme du gradient du Lagrangien s'annule et que les contraintes sont respectées.

Prise en compte de contraintes d'inégalité

On prend à présent en compte la présence d'un plancher convexe linéaire définit dans le plan (x,y) au moyen de la fonction affine :

$$\phi(x) = r + xs \qquad r \in \mathbb{R}^p \; s \in \mathbb{R}^p$$

Tous les points doivent se trouver dans l'ensemble convexe :

$$C = \{(x, y) \in \mathbb{R} \times \mathbb{R} : ye \geq \phi(x)\}$$

Avec $e := (1, 1, \ldots, 1)^T \in \mathbb{R}^p$
La chaîne étant affine entre ses noeuds, elle sera entièrement contenue dans C si tous les noeuds s'y trouvent, c'est à dire pour tout $i \in [\![1, n_n]\!]$ et tout $j \in [\![1, p]\!]$ on a :

$$c_{n_n+(j-1)n_n+i}(x, y) = r_j + x_i s_j - y_i \leq 0$$

On obtient donc un problème d'optimisation sous la forme :

$$\begin{cases} \min e(x, y) & \\ c_i(x, y) = 0, & i = 1, \ldots, n_b \\ c_i(x, y) \leq 0, & i = n_b + 1, \ldots, n_b + pn_n \end{cases}$$

On obtient alors un problème avec n_b contraintes d'égalité et $n_n p$ contraintes d'inégalité.

On modifie donc le simulateur décrit auparavant de sorte à lui donner un argument `lmi` en plus : un vecteur colonne de taille n_n contenant les multiplicateurs de lagrange associés aux contraintes d'inégalité.
Le simulateur retourne alors deux éléments de sortie en plus :

1. `ci` : un vecteur c_I de taille $n_n p$ contenant la valeur des contraintes d'inégalité
2. `ai` : la jacobienne des contraintes d'inégalité, une matrice c'_I de $n_n p$ lignes et $2n_n$ colonnes.

Ainsi on a :

$$c_I(x) = \underset{(pn_n \text{ termes})}{\left\{ \begin{bmatrix} r_1 \\ \vdots \\ r_p \\ \vdots \\ \vdots \\ r_1 \\ \vdots \\ r_p \end{bmatrix} \right.} + \begin{bmatrix} x_1 s_1 \\ \vdots \\ x_1 s_p \\ \vdots \\ \vdots \\ x_{n_n} s_1 \\ \vdots \\ x_{n_n} s_p \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_1 \\ \vdots \\ \vdots \\ y_{n_n} \\ \vdots \\ y_{n_n} \end{bmatrix}$$

dont on peut

$$c'_I = \begin{pmatrix} M_1 & A_1 \\ M_2 & A_2 \\ \vdots & \vdots \\ M_{n_n} & A_{n_n} \end{pmatrix}$$

Avec

$$M_1 = \begin{pmatrix} s_1 & 0 & \dots & 0 \\ s_2 & 0 & \dots & 0 \\ \vdots & & & \\ s_p & 0 & \dots & 0 \end{pmatrix} \qquad M_k = \begin{pmatrix} 0 & \dots & s_1 & 0 & \dots & 0 \\ 0 & \dots & s_2 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & s_p & 0 & \dots & 0 \end{pmatrix} \qquad M_{n_n} = \begin{pmatrix} 0 & \dots & 0 & s_1 \\ 0 & \dots & 0 & s_2 \\ \vdots & & & \vdots \\ 0 & \dots & 0 & s_p \end{pmatrix}$$

M_k des matrices à p lignes et n_n colonnes et $k \in [\![1, n_n]\!]$ la k-ième colonne de la matrice

$$A_1 = \begin{pmatrix} -1 & 0 & \dots & 0 \\ -1 & 0 & \dots & 0 \\ \vdots & & & \\ -1 & 0 & \dots & 0 \end{pmatrix} \qquad A_k = \begin{pmatrix} 0 & \dots & -1 & 0 & \dots & 0 \\ 0 & \dots & -1 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & -1 & 0 & \dots & 0 \end{pmatrix} \qquad A_{n_n} = \begin{pmatrix} 0 & \dots & 0 & -1 \\ 0 & \dots & 0 & -1 \\ \vdots & & & \vdots \\ 0 & \dots & 0 & -1 \end{pmatrix}$$

A_k des matrices à p lignes et n_n colonnes et $k \in [\![1, n_n]\!]$ la kième colonne de la matrice

Les contraintes étant affines, il n'y a pas besoin de modifier le hessien de lagrangien.

Algorithme d'Optimisation Quadratique Successive : Résultats

Testons ce nouveau solveur sur différents cas-tests :
Dans le cas-test 4a, le solveur du problème quadratique osculateur converge vers le minimum global contrairement au précédent qui ne trouvait au mieux qu'un point stationnaire.
Si on prend simplement $M_k = I$ au lieu de $M_k = H_k + E_k$, on perd la convergence de l'algorithme,

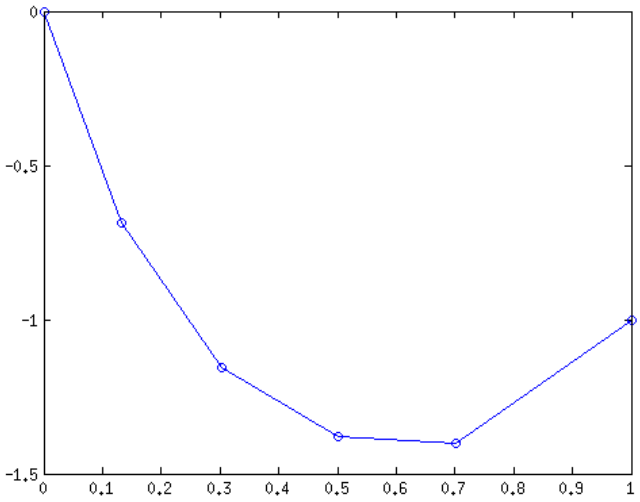
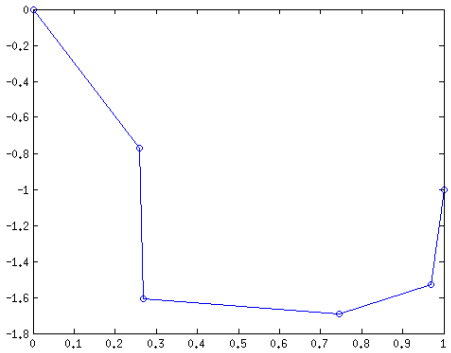
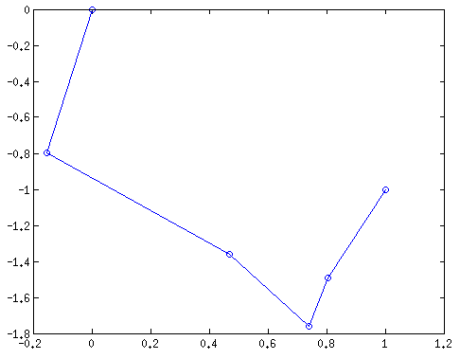


FIGURE 10 – Cas-test 4a avec optimisation quadratique successive

et en particulier dans le cas 4a, il oscille entre les deux positions de la Figure 11.



(a) position 1



(b) position 2

FIGURE 11 – Cas-test 4a avec $M_k = I$

Sur la Figure 12 est représentée la solution calculée pour le cas-test 4b, on voit que cette solution minimise l'énergie potentielle tout en restant au dessus du plancher.
Sur le cas 4c, l'algorithme converge bien vers le minimum et on peut observer sur la Figure 13

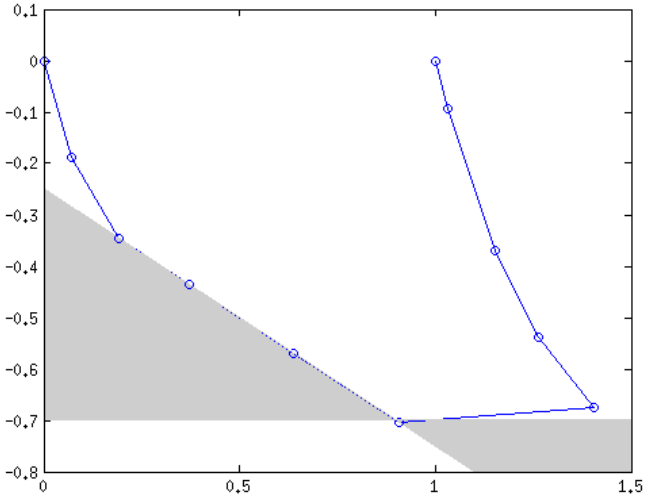


FIGURE 12 – Cas-test 4b

que les contraintes d'inégalité sont bien satisfaites. En effet la chaîne reste toujours au pire sur le plancher.
On peut aussi le vérifier numériquement en faisant tourner le solveur qui en mode bavard

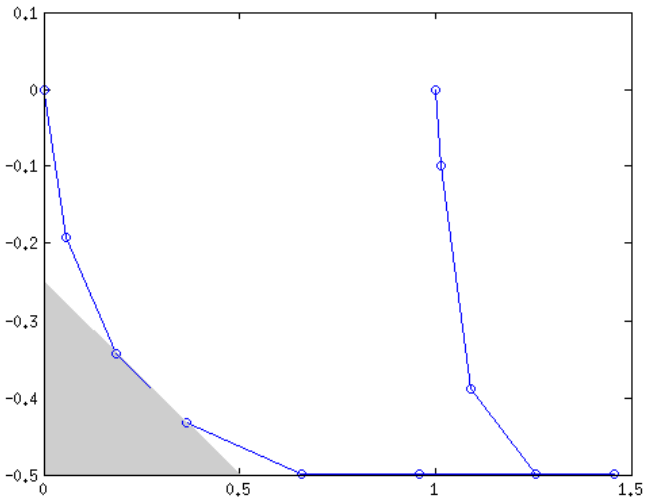


FIGURE 13 – Cas-test 4c

retournera un 1 si les contraintes d'inégalité sont satisfaites. La convergence de l'algorithme OQS est rapide, dans tous les cas on converge entre 18 et 30 itérations.