# Équilibre d'une chaîne articulée

BLANC Hugo
MAMFOUMBI Jean-Michel

Novembre 2020

## 1 Modélisation et écriture du simulateur

Dans cette première partie, nous allons introduire le problème étudié et les premières étapes mises en place afin de le résoudre. Aussi, nous traiterons donc de la modélisation du problème et de l'implémentation d'un simulateur associé au problème.

### 1.1 Définition du problème

Nous allons étudier le problème consistant à trouver la position d'équilibre statique d'une chaîne formée de barres rigides, contenue dans un plan vertical et fixée à ses deux bouts. On peut obtenir cette position en minimisant l'énergie potentielle de la chaîne.

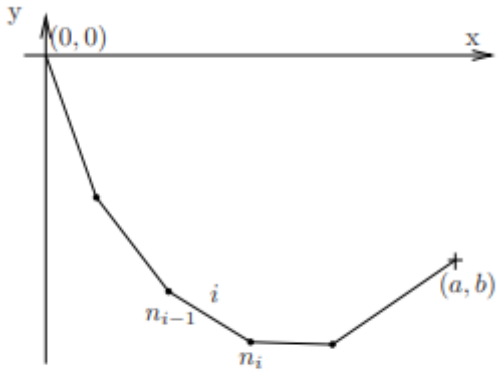



Figure 1: Chaîne formée de barres

Les coordonnées des noeuds (point d'articulation entre deux barres rigides) permettront de décrire la position de la chaîne. On note $n_b$ le nombres de barres et $n_n$ le nombre de noeuds.
De plus: $n_n = n_b - 1$ car les noeuds aux extrémités $(0, 0)$ et $(a, b)$ sont fixés et ne sont donc pas comptés dans les points d'articulations.
Le noeud $i$ sera noté dans le plan vertical par les coordonnées $(x_i, y_i)$, ces différentes coordonnées des points d'articulation seront donc les $2n_n$ variables de notre problème. Les deux extrémités fixées seront notées $(x_0, y_0) = (0, 0)$ et $(x_{n_b}, y_{n_b}) = (a, b)$.
De plus, la barre $i$ liée aux extrémités $i - 1$ et $i$ a pour longueur:

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

Et pour qu'une valeur quelconque $(x, y)$ soit admissible, il faut donc qu'elle vérifie les contraintes suivantes sur la longueur des barres, où la longueur de la barre i est noté $L_i$ :

$$\forall i \in \{1, ...n_b\},\ c_i(x, y) = 0$$

tel que $c_i(x, y) = l_i(x, y)^2 - L_i^2$.
On a passé les longueurs au carré afin que les contraintes soient différentiables. Enfin, on notera l'énergie potentielle de la barre $E$, c'est cette quantité qui doit être minimisé sous contraintes afin de résoudre le problème et d'après l'étude réalisé préalablement dans l'énoncé, on peut se ramener à la minimisation de la quantité e suivante:

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

sous les contraintes données précédemment.
Ainsi, on se rammène à la résolution du problème suivant:

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

## 1.2 Objectif du projet

L'objectif du projet est d'écrire un simulateur sur Matlab et un optimiseur pour le problème $(\mathcal{P})$, consistant à trouver la position d'équilibre de la chaîne.
De plus, en reprenant la formulation introduite précédemment et afin de fixer définitivement les notations on peut réécrire le problème $(\mathcal{P})$ sous la forme suivante:

$$(\mathcal{P}) \begin{cases} \min e(x) \\ c_i(x) = 0, \quad i \in \{1, ..., n_b\}, \end{cases}$$

tel que $x \in \mathbb{R}^n$, où $x$ correspond au couple (x,y) que nous utilisions précédemment, de plus on a donc $n = 2n_n$ .

## 1.3 Implémentation sur Matlab

Désormais, l'objectif du projet est clairement défini et les notations sont introduites. Nous allons donc pouvoir implémenter un programme principal noté *ch* qui contiendra les données du problème et fera l'appel à notre simulateur *chs*, qui calculera *e,c,g,a* et *hl* (qui ont été définie dans l'intitulé du projet) selon la valeur d'entrée que nous lui aurons indiqué (valeur de *indic*). Ainsi, une fois l'implémentation de *ch* réalisé, nous obtenons le code suivant:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                Projet OPT201-OPT202                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Pilotage du solveur

% Variables pilotant le solveur

indic=4;

% indic = 1 : chs fait un tracé de la chaîne (fonction plot);
% indic = 2 : chs calcule e et c;
% indic = 4 : chs calcule e, c, g et a;
% indic = 5 : chs calcule hl.

%% Définition des variables du problème

%Coordonnée du point de fixation extrème
global A; A = 1;
global B; B=-1;

%Longueur des barres et taille du vecteur associé

global L; L=[0.7;0.5;0.3;0.2;0.5];
global Nb; Nb=length(L);

% Coordonnées des noeuds non fixés
xy=[0.2;0.4;0.6;0.8;-1.0;-1.5;-1.5;-1.3];
xy
global N; N=length(xy);

lm=[];%A calculer

%%Appel au solveur

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

Figure 2: Implémentation de ch

Aussi, on peut remarquer que l'on a introduit les différentes données du problème: les coordonnées aux extrémités, les différentes longueurs des barres, le nombres de barres, le nombre de variables, les multiplicateurs de Lagrange pour les contraintes d'égalité ainsi que la la valeur courante de notre variable x qui est noté ici xy (dans l'implémentation).
De plus, *indic* permettra de piloter le fonctionnement du simulateur(cf figure 2).
Nous passons maintenant à l'implémentation du simulateur que l'on va construire progressivement selon les valeurs de *indic*:

- Pour $indic = 1$, nous traçons la chaîne, un appel à la fonction *plot* permet de facilement réaliser cela:

```
% On liste les abcisses et les coordonnées des noeuds
% dans deux vecteurs distincts ce qui sera utile pour
% afficher la chaîne
x=[0; xy(1:N/2); A];
y=[0; xy((N/2+1):N);B];
if indic==1
  plot(x,y);
  indic=0;
```

De plus, l'exécution du simulateur avec cette valeur de *indic* nous renvoie la représentation de la chaîne suivante:

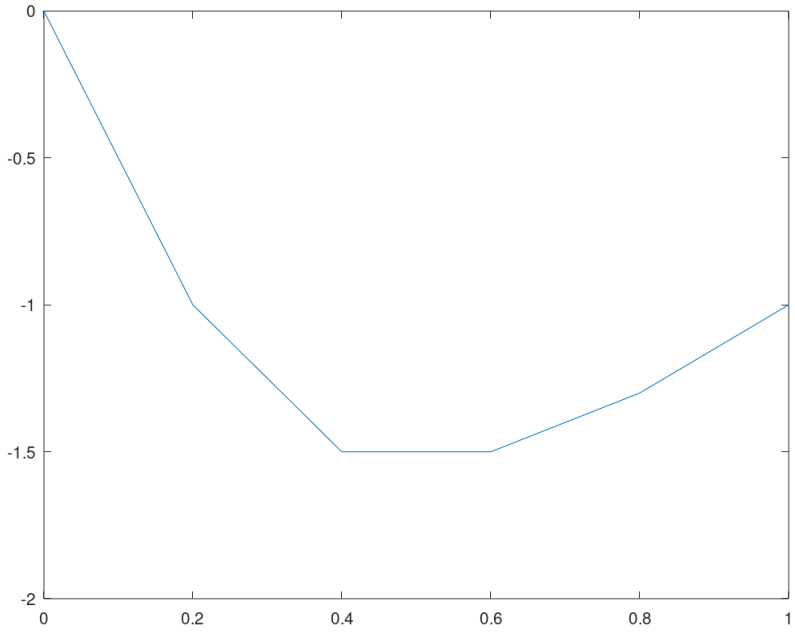



Figure 3: Représentation de la chaîne

Passons maintenant à l'implémentation de la suite du simulateur.

- Pour *indic*= 2, l'appel à notre simulateur doit permettre de calculer $e$ et $c$, et l'on rappelle que $e$ correspond à la valeur de l'énergie potentielle en xy et $c$ est la valeur en xy des contraintes sur la longueur des barres. Aussi, il suffit d'utiliser les formules que nous avions introduit dans la définition du problème et l'on obtient l'implémentation suivante:

```
elseif indic==2
  y_i=y(2:Nb+1);
  y_j=y(1:Nb);
  x_i=x(2:Nb+1);
  x_j=x(1:Nb);
  e=(L')*(y_i+y_j)/2;
  c=(x_i-x_j).^2+(y_i-y_j).^2-L.^2;
  indic=0;
```

Etudions désormais le comportement voulu du simulateur pour la valeur suivante de *indic*.

- Pour *indic*=4 en entrée, nous désirons calculer $e$, $c$, $g$ et $a$. Le calcul de $e$ et de $c$ a déjà été abordé. De plus, $g$ est le gradient de $e$ en xy et $a$ est la jacobienne des contraintes d'égalité en xy. Calculons ces deux quantitées, pour cela déterminous les dérivées partielles de $e$ et $c$, on a:

$$\begin{aligned} e(x,y) &= \sum_{i=1}^{n_b} L_i \frac{y_i + y_{i-1}}{2} \\ &= \sum_{i=1}^{n_b} L_i \frac{y_i}{2} + \sum_{i=0}^{n_b-1} L_{i+1} \frac{y_i}{2} \\ &= \sum_{i=1}^{n_b-1} y_i \frac{L_i + L_{i+1}}{2} + L_{n_b} \frac{y_{n_b}}{2} \qquad \text{(car } y_{n_b} \text{ est fixe et } y_0 = 0\text{)} \end{aligned}$$

Aussi on a $n_n = n_b - 1$, donc d'après l'expression précédente:

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

Ainsi, on peut désormais calculer $g$ donc les coordonnées sont les dérivées partielles précédente.
Passons maintenant à $a$, on a:

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

Donc: $\forall i \in \{1,...,n_b\}, \forall j \in \{1,...,n_n\}$,

$$\frac{\partial c_i}{\partial x_j}(x,y) = \begin{cases} 0 \text{ si } j \in \{1,...,n_n\}\backslash\{i-1,i\} \\ 2(x_i - x_{i-1}) \text{ si } j = i \\ 2(x_{i-1} - x_i) \text{ si } j = i-1 \end{cases}$$

et:

$$\frac{\partial c_i}{\partial y_j}(x,y) = \begin{cases} 0 \text{ si } j \in \{1,...,n_n\}\backslash\{i-1,i\} \\ 2(y_i - y_{i-1}) \text{ si } j = i \\ 2(y_{i-1} - y_i) \text{ si } j = i-1 \end{cases}$$

On peut donc calculer $a = \left(\frac{\partial c_i}{\partial (x,y)_j}(x,y)\right)_{1\leq i\leq n_b, 1\leq j\leq 2n_n}$ où $(x,y)_j$ est la j-ème composante de $(x,y) = \begin{pmatrix} x_1 \\ . \\ . \\ x_{n_n} \\ y_1 \\ . \\ . \\ y_{n_n} \end{pmatrix}$.

On implémente donc les différents calcul sous matlab:

```
elseif indic==4
  y_i=y(2:Nb+1);
  y_j=y(1:Nb);
  x_i=x(2:Nb+1);
  x_j=x(1:Nb);
  L_i=L(1:(Nb-1));
  L_j=L(2:Nb);
  e=(L')*(y_i+y_j)/2;
  c=(x_i-x_j).^2+(y_i-y_j).^2-L.^2;
  g=[zeros(Nb-1,1) ; (L_i+L_j)/2];
  a= [spdiags([2*(x_j-x_i)(2:Nb) 2*(x_i-x_j)(1:Nb-1)], -1:0, Nb,Nb-1),...
  spdiags([2*(y_j-y_i)(2:Nb) 2*(y_i-y_j)(1:Nb-1)], -1:0, Nb,Nb-1)];
  indic=0;
```

- Enfin pour *indic*= 5, on doit calculer hl, le hessien du lagrangien en xy et lm, pour cela il va falloir calculer les dérivées secondes de e et de chacun des $c_i$ afin de calculer leur hessien respectif.
On a d'après les calculs effectués à la question précédente:

$$\forall i \in \{1,...,2n_n\}, \forall j \in \{1,...,2n_n\}, \frac{\partial^2 e}{\partial (x,y)_i \partial (x,y)_j}(x,y) = 0$$

Donc, $\nabla^2 e((x,y)) = 0$.
De plus, $\forall i \in \{1,...,n_b\}, \forall j \in \{1,...,n_n\}$, $\forall k \in \{1,...,n_n\}$,

$$\frac{\partial^2 c_i}{\partial x_j \partial x_k}(x,y) = \begin{cases} 2 \text{ si } i = j = k \text{ ou si } i = j+1 = k+1 \\ -2 \text{ si } i = j = k+1 \text{ ou si } i = j+1 = k \\ 0 \text{ sinon} \end{cases}$$

et:

$$\frac{\partial^2 c_i}{\partial y_j \partial y_k}(x,y) = \begin{cases} 2 \text{ si } i = j = k \text{ ou si } i = j+1 = k+1 \\ -2 \text{ si } i = j = k+1 \text{ ou si } i = j+1 = k \\ 0 \text{ sinon} \end{cases}$$

et: $\frac{\partial^2 c_i}{\partial x_j \partial y_k}(x,y) = 0$
Aussi, nous sommes donc en mesure de calculer le hessien du lagrangien en xy et lm et de réaliser son implémentation:

```
elseif indic==5
  zero=zeros(Nb-1,Nb-1);
  aux=zero;
  aux(1,1)+=lm(1)*2;
  hl =[aux, zero;
       zero, aux;];
  for i=2:(Nb-1)
    aux=zero;
    aux(i,i)=2;
    aux(i-1,i-1)=2;
    aux(i-1,i-1)=2;
    aux(i-1,i)=-2;
    aux(i,i-1)=-2;
    hl+=lm(i)*[aux, zero;
       zero, aux;];
    end
  aux=zero;
  aux(Nb-1,Nb-1)=lm(Nb)*2;
  hl +=[aux, zero;
       zero, aux;];
```

Ainsi, nous avons désormais décrit le fonctionnement et l'implémentation de notre simulateur *chs*. Nous pouvons désormais passer à la vérification de l'exactitude des résultats renvoyés par le simulateur.

## 1.4 Vérification de l'exactitude du simulateur

### 1.4.1 Calcul exact des fonctions e et c sur des cas élémentaires:

Vérifions l'exactitude du calcul de nos fonctions e et c, si on se place par exemple sur le cas simple constitué de trois barres de longueurs égales respectivement à $[0.5, 1, 0.5]$ et avec deux points d'articulation en $(0.5, 0)$ et en $(0.5, -1)$ on est dans le cas suivant:

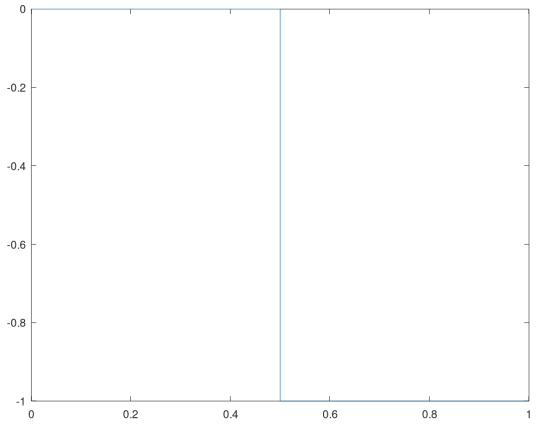



Figure 4: Représentation de la chaîne (Cas simple)

Nous devrions obtenir alors une énergie $e = -1$ J et des contraintes $c = 0$ d'après le calcul exact.
On obtient grâce à notre simulateur les résultats suivants:

```
c =

     0
     0
     0
```

```
e = -1
```

Figure 5: Résultat du simulateur (Cas simple)

Ce qui correspond donc à la solution exacte que nous attendions, ceci atteste de l'exactitude des valeurs e et c calculés par notre simulateur.

### 1.4.2 Exactitude des dérivées:

Pour déterminer l'exactitude des dérivées on reproduit la démarche introduite dans l'énoncé, ce qui après implémentation nous permet d'obtenir les dérivées calculées et les quotients différentiels suivants:

| i | deriv_e = | DF = |
|---|---|---|
| 1 | 0.00000 | 0.00000 |
| 2 | 0.00000 | 0.00000 |
| 3 | 0.00000 | 0.00000 |
| 4 | 0.00000 | 0.00000 |
| 5 | 0.60000 | 0.60000 |
| 6 | 0.40000 | 0.40000 |
| 7 | 0.25000 | 0.25000 |
| 8 | 0.35000 | 0.35000 |

Figure 6: Comparaison pour e des dérivées du simulateur
et des quotients différentiels associés (Cas-test)

Aussi, la première colonne donne l'indice de la composante du gradient, la seconde la composante du gradient calculé et la troisième les quotients différentiels correspondants. On peut remarquer que les solutions renvoyées sont similaires, nous vérifierons par la suite les erreurs relatives.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.4 | 0 | 0 | 0 | -2 | 0 | 0 | 0 |
| 2 | -0.4 | 0.4 | 0 | 0 | 1 | -1 | 0 | 0 |
| 3 | 0 | -0.4 | 0.4 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | -0.4 | 0.4 | 0 | 0 | -0.4 | 0.4 |
| 5 | 0 | 0 | 0 | -0.4 | 0 | 0 | 0 | -0.6 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.4 | 0 | 0 | 0 | -2 | 0 | 0 | 0 |
| 2 | -0.4 | 0.4 | 0 | 0 | 1 | -1 | 0 | 0 |
| 3 | 0 | -0.4 | 0.4 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | -0.4 | 0.4 | 0 | 0 | -0.4 | 0.4 |
| 5 | 0 | 0 | 0 | -0.4 | 0 | 0 | 0 | -0.6 |

Figure 7: Comparaison pour c des dérivées du simulateur
et des quotients différentiels associés (Cas-test)

Aussi, la première matrice donne la composante du jacobien calculé par le simulateur et la seconde les quotients différentiels correspondants. On peut remarquer que les solutions renvoyées sont similaires, nous vérifierons par la suite les erreurs relatives.
Après calcul, on obtient exclusivement des erreurs relatives inférieurs ou de l'ordre de $10^{-8}$ dans le cas de l'approximation des dérivées de e mais aussi pour l'approximation des dérivées de c (cf. implémentation au bas du fichier ch.m). Ainsi, ceci permet d'attester de la bonne exactitude des dérivées calculées par notre simulateur.

## 1.5 Etude théorique du problème (première partie-TP1)

**1)** Montrons que le problème (P) admet une solution s'il existe une chaîne admissible. Pour cela, on va appliquer le théorème de Weierstrass qui nous donne l'existence d'une solution pour le problème (P) à condition que l'ont ait :

- $e : X \mapsto \overline{R}$ est semi-continue inférieurement sur X (ensemble admissible)
- X compact non vide

Or, on a que e est continue donc semi-continue inférieurement, de plus s'il existe une chaîne admissible alors X (l'ensemble admissible) est non vide. Il nous reste donc à montrer que X est compact pour pouvoir appliquer le théorème. On remarque que $X \subset \mathbb{R}^n$ donc $X$ compact équivaut à $X$ fermé borné. Montrons maintenant ces deux propriétés :

- X est fermé car $X = \bigcap_{i=1}^{n} c_i^{-1}(\{0\})$ .On a $c_i^{-1}(\{0\})$ fermé en tant qu'image réciproque d'un fermé par une applicatoin continue pour tout i et $X$ est donc fermé en tant qu'intersection d'ensemble fermés.
- X est borné :
  On va raisonner par l'absurde et supposer que X n'est pas borné, alors il existe une suite $(x_n)_{n\in\mathbb{N}}$ dans $X$ telle que $|x_n| \underset{n\to+\infty}{\longrightarrow} +\infty$
  En dimension finie toutes les normes sont équivalentes donc on va considérer la norme $|x| = \max_{1\leq i\leq n}(|x_i|)$.
  On va montrer qu'il existe $n_0 \in \{1, ..., n\}$ et $\phi$ extractrice telle que $x_{n_0}^{\phi(n)} \underset{n\to+\infty}{\longrightarrow} +\infty$ ou $y_{n_0}^{\phi(n)} \underset{n\to+\infty}{\longrightarrow} +\infty$.
  Dans la suite de la démonstration on considèrera que c'est $x_{n_0}^{\phi(n)}$ (ça ne change rien au raisonnement)
  **Démonstration :** raisonnons par l'absurde et supposons que pour tout i dans $\{1, ..., n\}$ $(x_i^n)_{n\in\mathbb{N}}$ est borné. Alors on n'a pas $|x_n|$ tend vers $+\infty$ donc on aboutit à une contradiction sur notre hypothèse de départ, cela achève la démonstration de notre lemme (qu'on utilisera ci-dessous).
  Donc il existe bien $n_0 \in \mathbb{N}$ et $\phi$ telle que $x_{n_0}^{\phi(n)} \underset{n\to+\infty}{\longrightarrow} +\infty$.

$$\text{Or } c_{n_0}(x_n) = 0,\ \forall n \in \mathbb{N}$$

$$\implies (x_{n_0}^{\phi(n)} - x_{n_0-1}^{\phi(n)})^2 + (y_{n_0}^{\phi(n)} - y_{n_0-1}^{\phi(n)})^2 = l_{n_0}^2$$

$$\implies |x_{n_0}^{\phi(n)} - x_{n_0-1}^{\phi(n)}|^2 \leq l_{n_0}^2$$

$$\implies |x_{n_0}^{\phi(n)} - x_{n_0-1}^{\phi(n)}| \leq l_{n_0}$$

$$\implies |x_{n_0-1}^{\phi(n)}| \underset{n\to+\infty}{\longrightarrow} +\infty$$

  En raisonnant par récurrence descendante on obtient

$$|x_1^{\phi(n)}| \underset{n\to+\infty}{\longrightarrow} +\infty$$

  Or $x_1^n \leq l_1,\ \forall n \in \mathbb{N}$, donc $(x_1)_n$ est bornée et on aboutit à une contradiction, on ne peut donc pas construire une suite non bornée d'éléments de $X$ donc X est **borné**.
  On a donc $X$ fermé et borné en dimension finie donc compact. Donc on peut appliquer le théorème de Weierstrass et on a bien au moins une solution s'il existe une chaîne admissible.

**2)** Montrons que s'il existe $x_*$ solution de (P) alors il existe un multiplicateur optimal $\lambda_*$ tel que $\nabla_x l(x_*, \lambda_*) = 0$. Dans la question on note n le nombre de noeuds de notre chaine qui vont correspondre à nos variables d'optimisation $x_i, y_i$
Notre lagrangien est défini comme suit :

$$l(x, \lambda) = e(x) + \lambda^T c(x)$$

$$l(x, \lambda) = e(x) + \sum_{i=1}^{n_b} \lambda_i c_i(x)$$

D'où

$$\nabla_x l(x, \lambda) = \nabla_x e + \sum_{i=1}^{n_b} \lambda_i \nabla_x c_i(x)$$

Avec

$$\nabla_x c_i(x) = \begin{pmatrix} \frac{\partial c_i}{x_1} \\ ... \\ \frac{\partial c_i}{y_n} \end{pmatrix}$$

On peut écrire

$$\nabla_x l(x, \lambda) = \nabla_x e + A\lambda$$

où

$$A = (\nabla_x c_1(x), ..., \nabla_x c_{nb}(x))$$

Or si les contraintes sont qualifiées le théorème 4.17 (condition nécéssaire de Lagrange) nous donne que si $x_*$ est solution alors il existe $\lambda_*$ multiplicateur de lagrange optimal tel que

$$\nabla_x e + c'(x_*)^* \lambda_* = 0$$

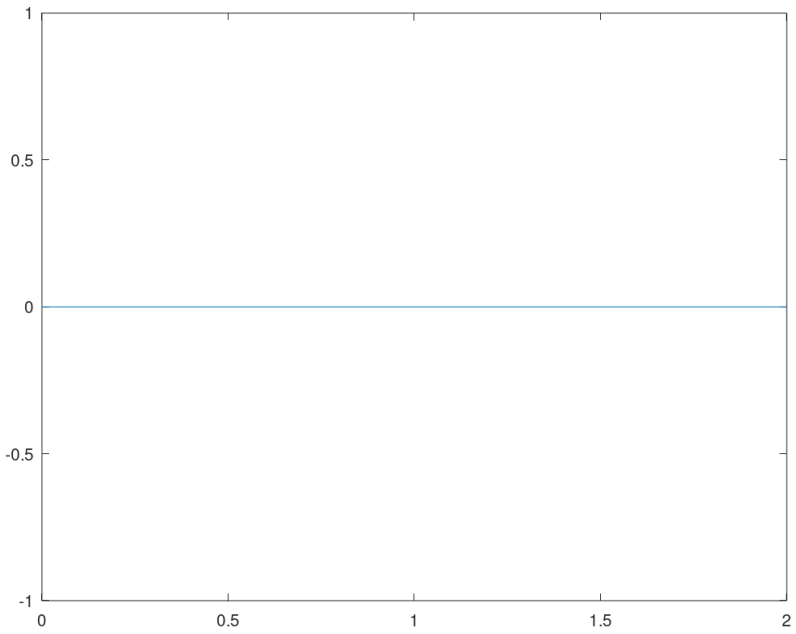
Avec $c'(x_*)^*$ l'opérateur adjoint de la jacobienne donc

$$c'(x_*)^* = \begin{pmatrix} \frac{\partial c_1}{\partial x_1} & ... & \frac{\partial c_{nb}}{\partial x} \\ ... & & ... \\ \frac{\partial c_1}{\partial y_n} & ... & \frac{\partial c_{nb}}{\partial y_n} \end{pmatrix} = A$$

Il nous reste à montrer que les contraintes sont qualifiées pour pouvoir appliquer le théorème et obtenir le résultat car on aura alors

$$\exists \lambda_* \in \mathbb{R}^{nb} : \nabla_x e + A\lambda_* = 0$$
$$\implies \exists \lambda_* \in \mathbb{R}^{nb} : \nabla_x l(x_*, \lambda_*) = 0$$

Montrons que les contraintes sont qualifiées. Une condition suffisante de qualification des contraintes est que $c'(x_*)$ est surjective.
Malheureusement, $c'(x_*)$ n'est pas nécéssairement surjectif, effectivement plaçons nous dans un cas simple afin d'illustrer un contre-exemple.
On prend $n_n = 1$, $n_b = 2$, $L_1 = 1$, $L_2 = 1$, $(x_0, y_0) = (0, 0)$, $(x_2, y_2) = (2, 0)$. La seule solution admissible étant donc optimale est donc $(x, y) = (x_1, y_1) = (1, 0)$, ce qui nous rammène à la situation suivante:




De plus, on a donc :

$$c'(x_*) = \begin{pmatrix} 2 & 0 \\ -2 & 0 \end{pmatrix}$$

qui n'est donc pas surjective (car de rang $1 < 2$), par théorème il n'y a donc pas nécessairement unicité du multiplicateur optimal $\lambda_*$ associé à $x_*$.

On n'a pas la surjectivité donc il faudrait revenir à la définition de la qualification des contraintes c'est à dire égalité du cône linéarisé et du cône tangent.

## 1.6 Application numérique

Testons maintenant notre simulateur pour un cas-test simple où l'on choisit

- 5 barres de longueur L = [0.7 0.5 0.3 0.2 0.5]'
- la poistion initiale des noeuds est xy = [ 0.2 0.4 0.6 0.8 -1.0 -1.5 -1.5 -1.3 ]'
- et notre deuxième point de fixation (a,b) = (-1,1)

On lance le simulateur pour indic = 1 on obtient bien le tracé de la chaîne pour les données indiquées :

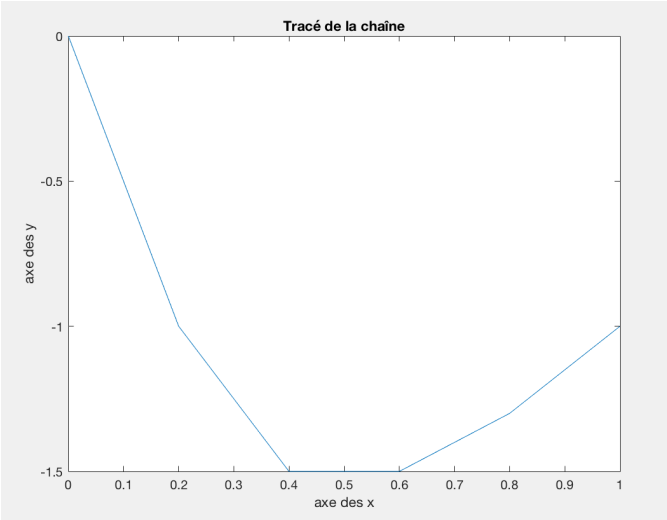



Figure 8 Tracé de la chaîne dans le cas test

Pour indic = 2, on obtient comme désiré le calcul des fonctions et c en xy:

```
e =

   -2.2800


c =

    0.5500
    0.0400
   -0.0500
    0.0400
   -0.1200
```

Figure 9 : Résultat du simulateur pour le cas indic = 2

Pour indic = 4, on obtient le calcul de e, c, g et a pour notre cas-test:

```
e =

   -2.2800


c =

    0.5500
    0.0400
   -0.0500
    0.0400
   -0.1200


g =

         0
         0
         0
         0
    0.6000
    0.4000
    0.2500
    0.3500


a =

    0.4000         0         0         0   -2.0000         0         0         0
   -0.4000    0.4000         0         0    1.0000   -1.0000         0         0
         0   -0.4000    0.4000         0         0         0         0         0
         0         0   -0.4000    0.4000         0         0   -0.4000    0.4000
         0         0         0   -0.4000         0         0         0   -0.6000
```

Figure 10 : Résultat du simulateur pour le cas indic = 4

Pour indic = 5 en testant avec lm = [1, 1, 1, 1, 1] on obtient :

```
hl =

     4    -2     0     0     0     0     0     0
    -2     4    -2     0     0     0     0     0
     0    -2     4    -2     0     0     0     0
     0     0    -2     4     0     0     0     0
     0     0     0     0     4    -2     0     0
     0     0     0     0    -2     4    -2     0
     0     0     0     0     0    -2     4    -2
     0     0     0     0     0     0    -2     4
```

Figure 11 : Résultat du simulateur pour le cas indic = 5

C'est la matrice que l'on doit récupérer pour le lm choisi donc le calcul de hl semble correct.

# 2 Méthode de Newton pour problème d'optimisation avec contraintes d'égalité

Dans cette seconde partie, nous allons introduire la méthode de Newton permettant une résolution de notre problème de minimisation.

## 2.1 Implémentation de l'optimiseur

Une fois que l'on a mis en place le simulateur on va implémenter l'optimiseur qui va nous permettre de résoudre notre problème. On s'appuiera sur **la méthode de Newton** qui s'applique dans le cadre d'un problème d'optimisation avec contraintes d'égalité. On va chercher un point stationnaire associé aux conditions d'optimalité du premier ordre (conditions de Lagrange) ce qui revient à résoudre :

$$\begin{cases} \nabla e(x_*) + c'(x_*)^T \lambda_* = 0 \\ c(x_*) = 0 \end{cases}$$

On introduit pour cela $F : \mathbb{R}^N \mapsto \mathbb{R}^N$ définie comme suit :

$$F(z) = \begin{pmatrix} \nabla e(x) + c'(x)^T \lambda \\ c(x) \end{pmatrix} \text{ et } z = (x, \lambda) \in \mathbb{R}^N$$

et on cherche à résoudre $F(z_*) = 0$

Pour résoudre $F(z_*) = 0$ on va partir d'un $z = (x, \lambda)$ de départ qui sera notre $z_0 = (x_0, \lambda_0)$ et on va chercher à converger vers un point stationnaire. Tant que le $z_k$ actuel n'est pas un point stationnaire c'est-à-dire

$$\nabla l(x_k, \lambda_k) = 0 \text{ et } c(x_k) = 0$$

On définit $z_{k+1} = (x_{k+1}, \lambda_{k+1})$ comme suit :

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

La méthode de Newton nous donne la relation

$$F'(z_k) \begin{pmatrix} d_k \\ \lambda_k^{PQ} \end{pmatrix} = - \begin{pmatrix} \nabla e(x_k) \\ c(x_k) \end{pmatrix}$$

avec

$$F'(z_k) = \begin{pmatrix} \nabla^2_{xx} l(x_k, \lambda_k) & c'(x_k)^T \\ c'(x_k) & 0 \end{pmatrix}$$

Une implémentation possible sous Matlab est présentée dans la figure ci-dessous :

```
%%DEMARRAGE DE LA BOUCLE
while ((k < options.maxit) & ((norm(grad_l) > options.tol(1)) & (norm(c) > options.tol(2))))
   k = k+1
   epsilon = -F_prime\F ;

   %MIS A JOUR DE Z
   z = [x; zeros(m,1)] + epsilon ;
   x = z(1:n) ;
   lm = z(n+1:n+m);

   %MIS A JOUR DE F
   [e,c,g,A,hl] = simul(4,x) ; %on recupere e l'energie potentielle ,c vecteur contraintes, g gradient de e ,a jacobienne de c
   F = [g ; c] ;

   grad_l = g + A'*lm ;

   %MIS A JOUR DE F' ;
   [e1,c1,g1,A1,hl] = simul(5,x,lm) ; %on recupere hl le hessien de l
   F_prime = [hl A' ; A zeros(m,m)] ;
end
```

Figure 12 : Implémentation de la méthode de Newton

L'implémentation se décompose en plusieurs parties :

- définition des conditions d'arrêt de l'algorithme : à partir de l'entrée options on a accès aux seuils de tolérance pour lesquels on considère que l'optimum est atteint. L'algorithme s'arrête si on a :

$$||\nabla_x l(x_k, \lambda_k)||_\infty < \text{options.tol(1)} \text{ et } ||c(x_k)||_\infty < \text{options.tol(2)}$$

  ou si le nombre maximal d'itérations est dépassé.

- Calcul de $d_k$, $\lambda_k^{PQ}$ dans la boucle while
- Calcul de $z_{k+1}$ et mise à jour de $F$ et $F'$ et de $k$ qui donne le nombre d'itérations
- Vérification des conditions d'arrêt

## 2.2 Application numérique avec les cas-tests

On étudie 4 différents cas-test pour lesquels les noeuds ont des positions initiales différentes, ces noeuds sont associés à 5 barres de longueur 0.7, 0.5, 0.3, 0.2 et 0.5.
La tolérance (options) sera prise à $10^{-3}$ pour le calcul d'une solution et on prendra un nombre maximal d'itérations égal à 1000.

- Cas test 2a : (a,b) = (1,-1) et on prend comme position initiale des nœuds : (0.2, 1), (0.4, -1.5), (0.6, -1.5) et (0.8, -1.3). On obtient alors la position initiale suivante:

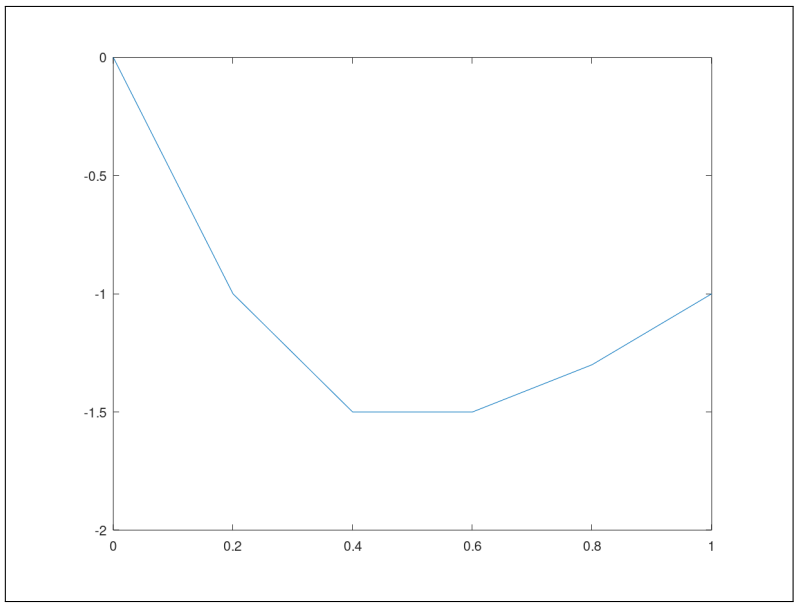



Figure 13 : Position initiale de la chaîne (cas test 2a)

La position optimisée est la suivante:

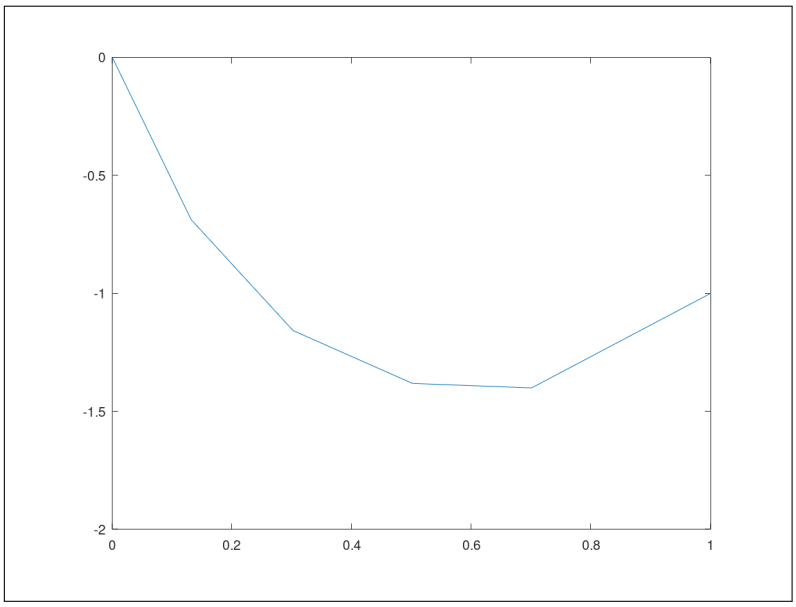



Figure 14 : Position optimisée de la chaîne (cas test 2a)

Pour plus de précision, le programme nous renvoie les vecteurs de positions des noeuds et des multiplicateurs lagrangiens (xy et lm) suivants:

```
x =

    0.13170
    0.30201  lm =
    0.50164
    0.70081     0.92604
   -0.68751     0.71611
   -1.15762     0.61010
   -1.38172     0.61217
   -1.40063     0.40767
```

Aussi, la convergence vers la solution se fait en 4 itérations avec le seuil d'optimalité atteint:

| info | |
|---|---|
| | Values |
| niter | 4 |
| status | 0 |

Pour connaître la nature du point de convergence, on calcule une base de vecteur propre de $\mathrm{Ker}(c')$, puis l'on concatène les différents vecteurs propres dans une matrice $K$, enfin l'on calcule les valeurs propres de la matrice $K^T L K$, où $L = \nabla^2_{xx} l(x_*, \lambda_*)$ ce qui nous permet de conclure sur la nature du point de convergence (cf annexe 1)):

```
[~,~,~,~,L,~]=chs(5,x,lm);
K=null(a);
[vecteurs_propres,valeurs_propres]=eig((K')*L*K);
valeurs_propres
```

Ici, il y a convergence de l'algorithme vers un **minimum local strict** car les valeurs propres de la matrice $K^T L K$ (hessienne réduite) sont strictement positives:

```
   4.54112         0         0
         0   0.75595         0
         0         0   2.53282
```

Le spectre de la matrice correspond ici à ses coefficients diagonaux.
Étudions désormais plus en détail l'étude de la convergence de la suite:

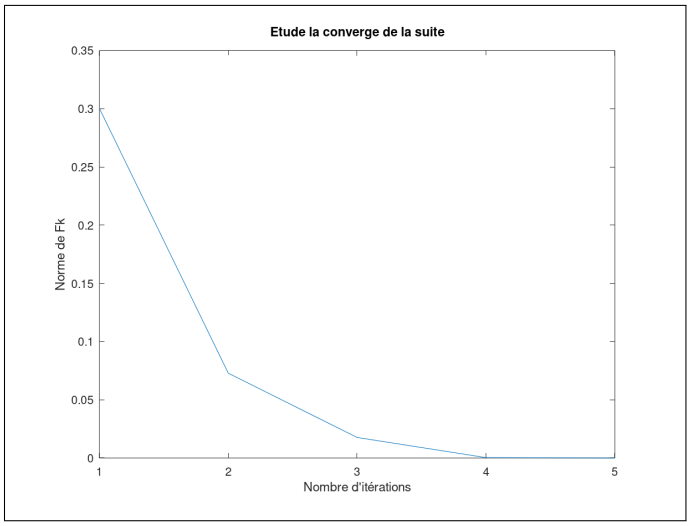



Figure 15: Étude de la convergence de la solution (cas test 2a)

On a tracé l'évolution de $||F(z_k)||_\infty$ à chaque itération, en effet cette quantité est adéquate pour étudier la vitesse de convergence de l'algorithme d'après l'étude théorique. On peut voir ici que la convergence est très rapide (la tolérance a été augmenté à $10^{-6}$ pour ce test).
Étudions désormais la convergence quadratique.

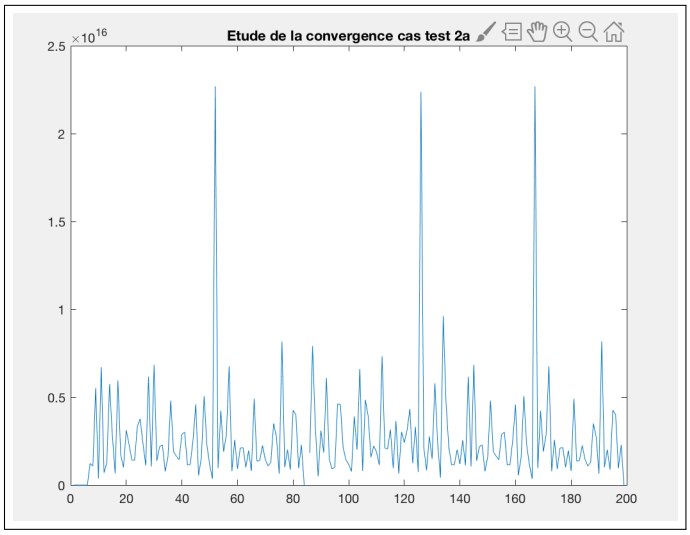



Figure 16: Étude de la convergence quadratique de la solution (cas test 2a)

On a calculé le rapport suivant pour $k \geq 1$

$$\frac{||z_{k+1} - z^*||}{||z_k - z^*||^2}$$

et on le trace pour 200 itérations. Si on a convergence quadratique alors on devrait avoir une suite de réels bornée par une constante $C > 0$ qu'on peut déterminer a posteriori. Après tracé on obtient la constante C, elle est très élevée, on suppose que c'est due aux erreurs d'approximation car on travaille avec des valeurs très proche de 0. Toutefois, on obtient tout de même que le quotient est borné, ainsi il y a convergence quadratique de la suite $(z_k)_k$ ce qui correspond au résultat théorique connu. En effet, on observe la convergence rapide vers la solution d'équilibre proche de notre position de départ comme cela serait attendu théoriquement.

- Cas test 2b : on prend comme position initiale des nœuds : (0.2, 1), (0.4, 1.5), (0.6, 1.5) et (0.8, 1.3).
On obtient la représentation initiale suivante de la chaîne:

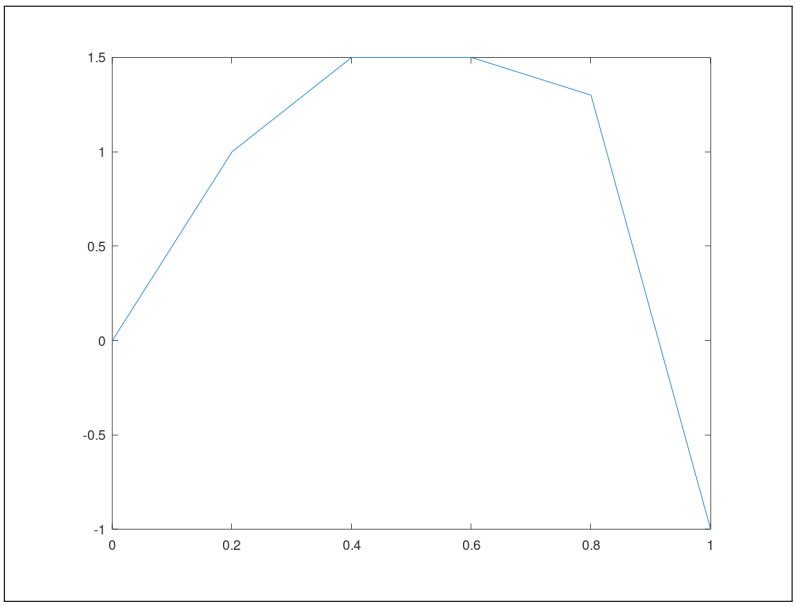



Figure 17 : Position initiale de la chaîne (cas test 2b)

Par ailleurs, la représentation de la chaîne optimisée est la suivante:

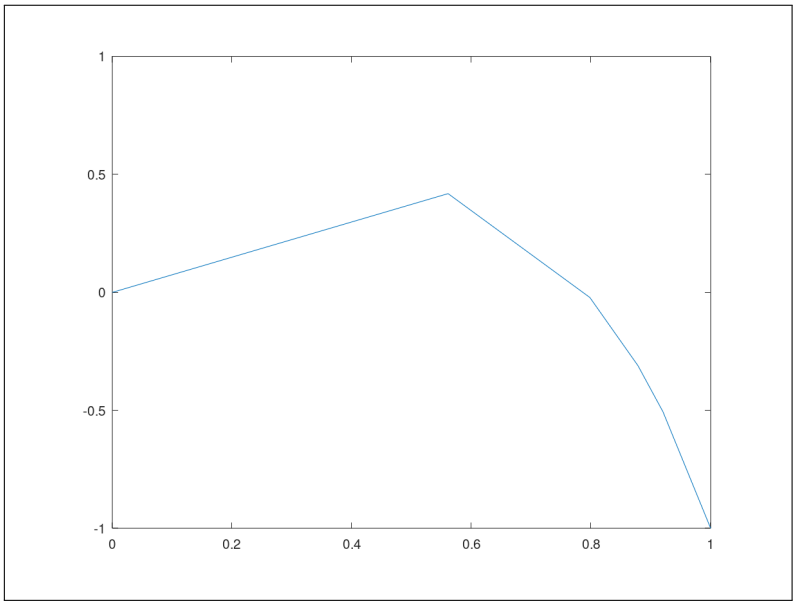



Figure 18 : Position optimisée de la chaîne (cas test 2b)

Pour plus de précision, le programme nous renvoie les vecteurs de positions des noeuds et des multiplicateurs lagrangiens (xy et lm) suivants:

```
x =

   0.561134
   0.798126
   0.878548
   0.920351
   0.418489
  -0.021782
  -0.310802
  -0.506385

lm =

  -0.20534
  -0.48620
  -1.43267
  -2.75621
  -1.44661
```

La convergence vers la solution se fait en 7 itérations avec le seuil d'optimalité atteint:

info

| | Values |
|---|---|
| niter | 7 |
| status | 0 |

Pour connaître la nature du point de convergence, on calcule une base de vecteur propre de $\mathrm{Ker}(c')$, puis l'on concatène les différents vecteurs propres dans une matrice $K$, enfin l'on calcule les valeurs propres de la matrice $K^T LK$, où $L = \nabla^2_{xx} l(x_*, \lambda_*)$ ce qui nous permet de conclure sur la nature du point de convergence (cf annexe 1)):

```
[~,~,~,~,L,~]=chs(5,x,lm);
K=null(a);
[vecteurs_propres,valeurs_propres]=eig((K')*L*K);
valeurs_propres
```

Ici, il y a convergence de l'algorithme vers un **maximum local strict**(ce qui coïncide avec la théorie car notre position initiale est proche de cet équilibre) car les valeurs propres de la matrice $K^T LK$ (hessienne réduite) sont strictement négatives:

```
-14.2644         0         0
       0   -5.0431         0
       0         0   -1.0437
```

Le spectre de la matrice correspond ici à ses coefficients diagonaux.
Étudions désormais plus en détail l'étude de la convergence de la suite:

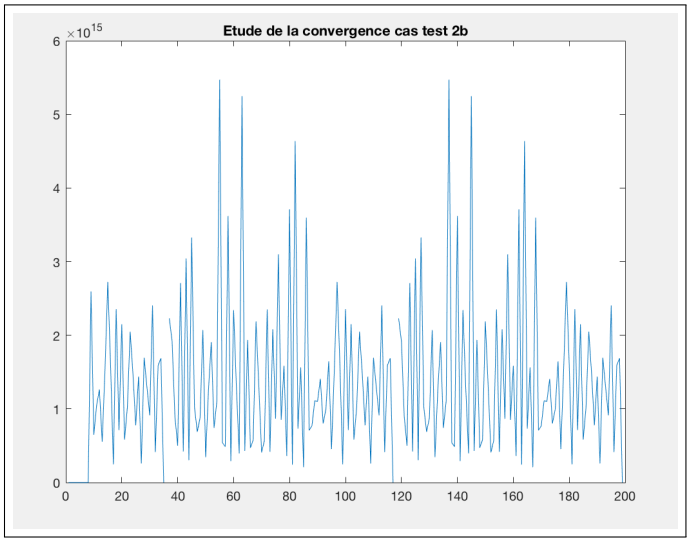



Figure 19: Étude de la convergence quadratique de la solution (cas test 2b)

On a calculé le rapport suivant pour $k \geq 1$

$$\frac{||z_{k+1} - z^*||}{||z_k - z^*||^2}$$

et on le trace pour 100 itérations. Si on a convergence quadratique alors on devrait avoir une suite de réels bornée par une constante $C > 0$ qu'on peut déterminer a posteriori. Après tracé on obtient la constante C, elle est très élevée, on suppose que c'est due aux erreurs d'approximation car on travaille avec des valeurs très proche de 0. Toutefois, on obtient tout de même que le quotient est borné, ainsi il y a convergence quadratique de la suite $(z_k)_k$, ce qui correspond aux attentes théoriques. En effet, on observe la convergence rapide vers la solution d'équilibre proche de notre position de départ (qui est ici un maximum).

- Cas test 2c : on prend comme position initiale des nœuds : (0.2, -1), (0.4, -1.5), (0.6, 1.5) et (0.8, -1.3).
On obtient la représentation initiale suivante de la chaîne:

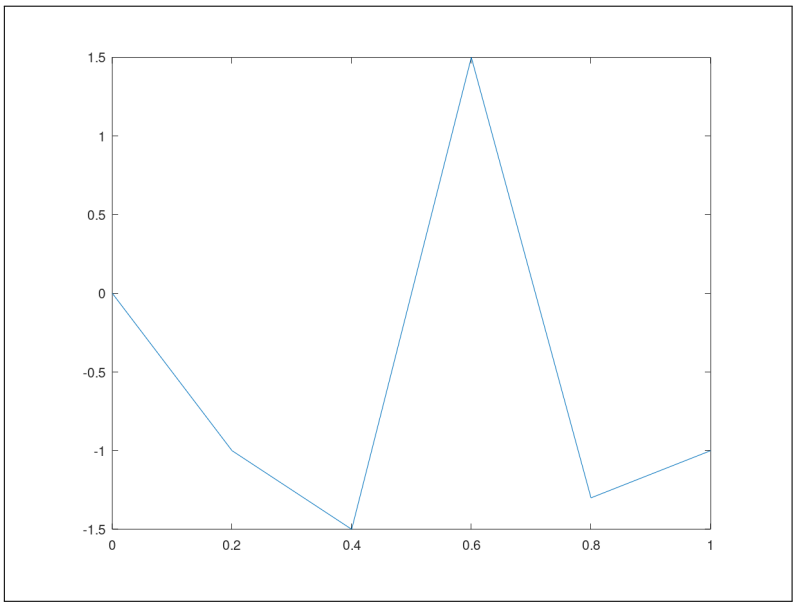



Figure 20 : Position initiale de la chaîne (cas test 2c)

Par ailleurs, la représentation de la chaîne optimisée est la suivante:

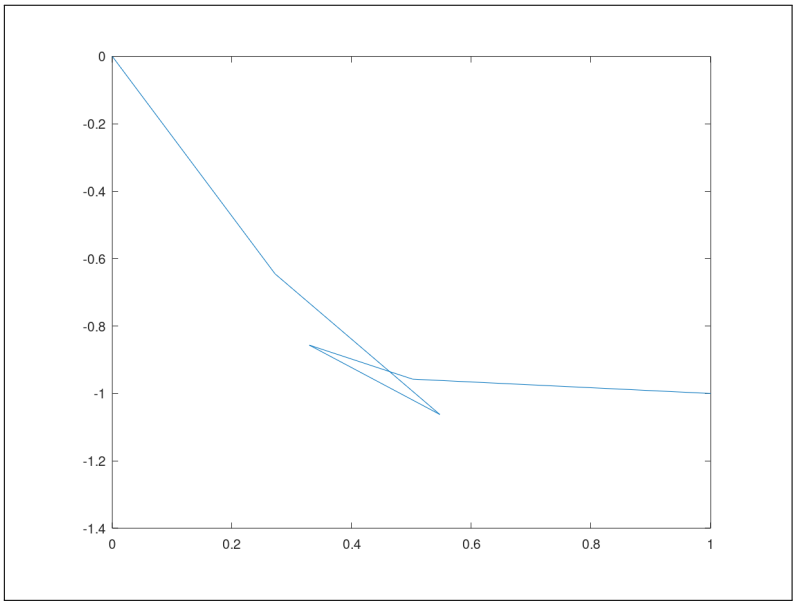



Figure 21 : Position optimisée de la chaîne (cas test 2c)

Pour plus de précision, le programme nous renvoie les vecteurs de positions des noeuds et des multiplicateurs lagrangiens (xy et lm) suivants:

```
x =

   0.27190
   0.54737
   0.32911
   0.50179
  -0.64504
  -1.06231
  -0.85648
  -0.95746

lm =

   1.28650
   1.26978
  -1.60259
   2.02559
   0.70210
```

La convergence vers la solution se fait en 11 itérations avec le seuil d'optimalité atteint:

info

| | Values |
|---|---|
| niter | 11 |
| status | 0 |

Pour connaître la nature du point de convergence, on calcule une base de vecteur propre de $\mathrm{Ker}(c')$, puis l'on concatène les différents vecteurs propres dans une matrice $K$, enfin l'on calcule les valeurs propres de la matrice $K^T LK$, où $L = \nabla^2_{xx} l(x_*, \lambda_*)$ ce qui nous permet de conclure sur la nature du point de convergence (cf annexe 1)):

```
[~,~,~,~,L,~]=chs(5,x,lm);
K=null(a);
[vecteurs_propres,valeurs_propres]=eig((K')*L*K);
valeurs_propres
```

Ici, il y a convergence de l'algorithme vers un **point selle** car les valeurs propres de la matrice $K^T LK$ (hessienne réduite) sont strictement négatives et strictement positives:

```
   -3.6139         0         0
         0    6.6495         0
         0         0    5.2384
```

Le spectre de la matrice correspond ici à ses coefficients diagonaux.
Cette convergence vers un point selle peut être dû au choix de notre position initiale qui peut être trop éloigné de la solution minimale, ainsi toutes les hypothèses théoriques de convergence vers la solution ne sont pas vérifiées.
Étudions désormais plus en détail l'étude de la convergence de la suite:

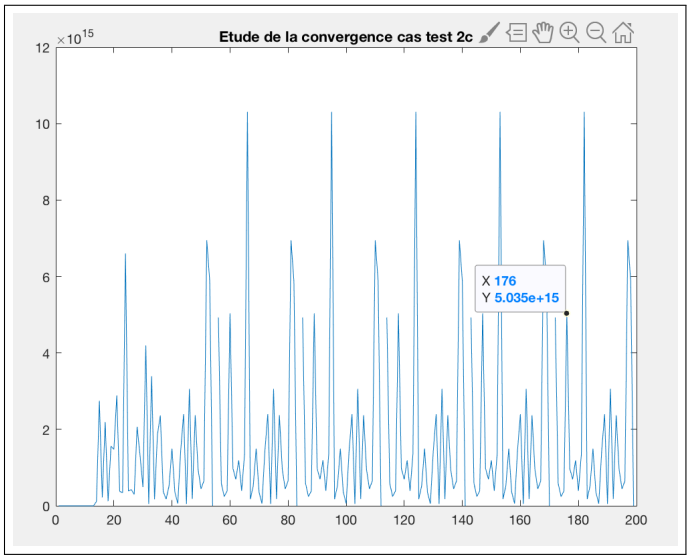



Figure 22: Étude de la convergence quadratique de la solution (cas test 2c)

On a calculé le rapport suivant pour $k \geq 1$

$$\frac{||z_{k+1} - z^*||}{||z_k - z^*||^2}$$

et on le trace pour 100 itérations. Si on a convergence quadratique alors on devrait avoir une suite de réels bornée par une constante $C > 0$ qu'on peut déterminer a posteriori. Après tracé on obtient la constante C, elle est très élevée, on suppose que c'est due aux erreurs d'approximation car on travaille avec des valeurs très proche de 0. Toutefois, on obtient tout de même que le quotient est borné, ainsi il y a convergence quadratique de la suite $(z_k)_k$.

- Cas test 2d : on prend comme position initiale des nœuds : (0.2, 1), (0.4, -1.2), (0.6, 1.5) et (0.8, -1.3).
On obtient dans ce dernier cas la représentation initiale suivante de la chaîne:

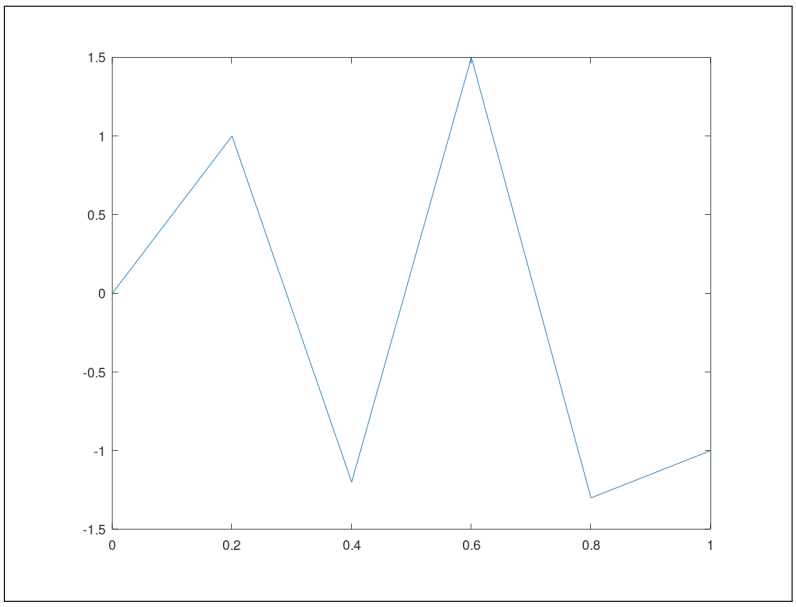



Figure 23 : Position initiale de la chaîne (cas test 2d)

Par ailleurs, la représentation de la chaîne optimisée est la suivante:

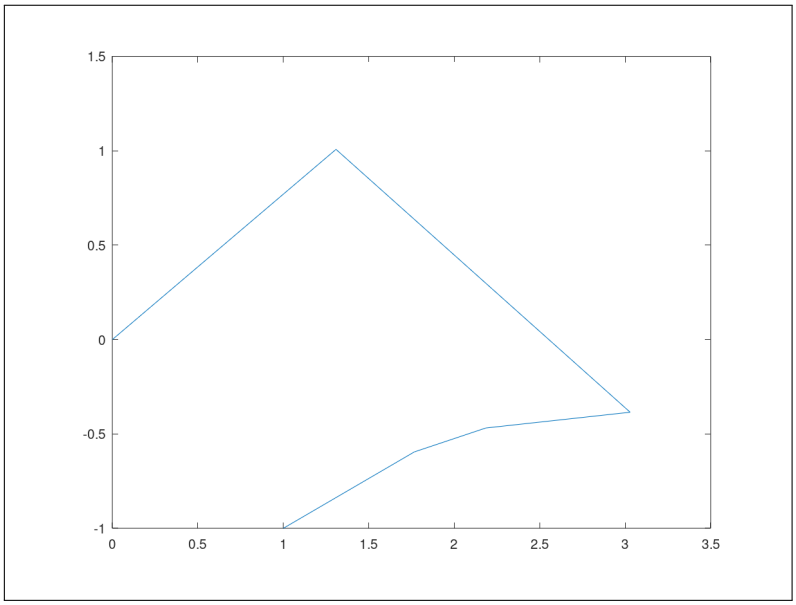
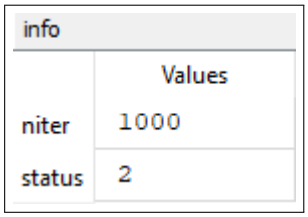



Figure 24 : Position optimisée de la chaîne (cas test 2d)

Il n'y a pas convergence, en effet nous atteignons le seuil d'itérations maximal (1000) et le statut indique que le seuil d'optimalité n'est pas atteint:




Étudions désormais plus en détail l'étude de la convergence de la suite:

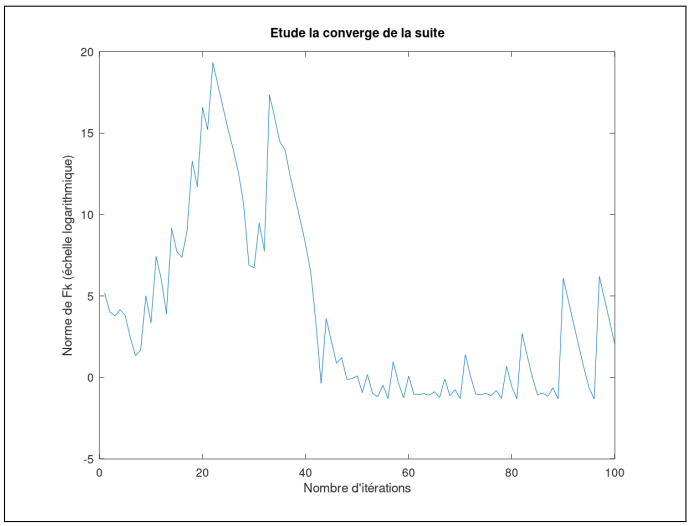



Figure 25: Étude de la convergence de la solution (cas test 2d)

On a calculé la quantité suivante pour $k \geq 1$:

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

et on la trace pour 100 itérations. Si on avait convergence de l'algorithme alors on devrait avoir une suite de réels tendant vers 0 d'après l'étude théorique. Toutefois, on obtient ici que cette norme ne tend pas vers 0, en effet comme on peut voir sur le graphe la quantité ne se stabilise pas vers 0. Ainsi il n'y a pas convergence de la suite $(z_k)_k$. La non-convergence peut être du au fait que notre position initiale est trop éloignée de toute solution stationnaire. En effet, lorsque on observe la position initiale on s'aperçoit qu'elle correspond à une position intermédiaire entre le cas a) et b).

## 2.3 Étude théorique de la convergence (seconde partie-TP2)

**1)** Nous allons étudier comment évaluer la vitesse de convergence de $z_k$ vers sa limite $z_*$ en examinant le comportement de:

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

On peut tout d'abord utiliser le fait que dans le cas où nous somme dans les conditions suivantes:

(a) $x_* \in \mathbb{E}$ est tel que $F(z_*) = 0$

(b) F est $C^{1,1}$ dans un voisinage de $z_*$

(c) F'($z_*$) est inversible

Alors, $\exists V$ un voisinage de $z_*$ tel que en prenant $z_1 \in V$, l'algorithme de Newton est bien défini et la suite généré par l'algorithme de Newton vérifie:

$$z_k \underset{k \to +\infty}{\longrightarrow} z_* \quad \textbf{quadratiquement}$$

Aussi, dans ses hypothèses on a la convergence vers $z_*$ de la suite $(z_k)_k$ généré par l'algorithme de Newton. De plus, sous ses hypothèses on a aussi par différentiabilité de F en $x_*$ que:

$$F(z) = F(z_*) + F'(z_*).(z - z_*) + o(||z - z_*||)$$

Aussi, la nullité en $z_*$, nous permet d'obtenir:

$$F(z) = F'(z_*).(z - z_*) + o(||z - z_*||)$$

On peut donc en déduire que:

$$||F(z)|| = O(||z - z_*||) \quad \text{au voisinage de } z_*$$

De plus, on a grâce à l'inversibilité de $F'(x_*)$, on a que:

$$z - z_* = F'(z_*)^{-1} F(z) + o(||z - z_*||)$$

Donc, on en conclut que:

$$||z - z_*|| = O(||F(z)||) \quad \text{au voisinage de } z_*$$

Ainsi, on en déduit finalement que:

$$||z - z_*|| \sim ||F(z)|| \quad \text{au voisinage de } z_*$$

Ici la norme $||.||$ désigne $||.||_\infty$ mais cela n'importe pas car en dimension finie, toutes les normes sont équivalentes.
On a donc montré que:
$\exists C \geq 1$ tel que $\forall z$ dans un voisinnage de $z_*$, on a:

$$C^{-1}||F(z)|| \leq ||z - z_*|| \leq C||F(z)||$$

Aussi, par convergence de la suite $(z_k)$ vers $z_*$, on a donc:

$$||z_k - z_*|| \sim_{k\to\infty} ||F(z_k)||$$

Aussi, on peut donc en déduire grâce à cet encadrement (obtenue grâce à l'équivalence précédente) que l'on peut évaluer la vitesse de convergence de $z_k$ vers sa limite $z_*$ en examinant le comportement de $||F(z_k)||_\infty$ à chaque itération (en prenant un point initial au voisinnage de $z_*$).
De plus, l'intérêt de cette méthode utilsant $||F(z_k)||_\infty$ à la place de $||z_k - z_*||$ est que $||F(z_k)||_\infty$ est calculable à chaque itération facilement car $z_k$ et la fonction $F$ sont connues, tandis que $z_*$ est parfois inconnue lors de l'utilisation de l'algorithme de Newton, il est donc plus complexe, difficile de calculer $||z_k - z_*||$.

**2)** Déterminons un coefficient $\lambda_1$ intéressant avec lequel débuter lorsque l'on calculera pour la première fois le hessien du lagrangien du **sqp** dans le cas où $x_1$ est un point stationnaire du problème.
La méthode de Newton se termine lorsque l'on a dépassé le nombre maximal d'itérations ou que l'on est sous les deux seuils de tolérance fixés au préalable. On sait que ces deux seuils portent sur $||c(x_1)||_\infty$ et $||\nabla_x l(x_1, \lambda_1)||_\infty$ .
Si $x_1$ est un point stationnaire du problème alors il vérifie $c(x_1) = 0$ donc la première condition est vérifiée. Pour que l'algorithme n'itère pas, il faut et il suffit qu'on ait $||\nabla_x l(x_1, \lambda_1)||_\infty \approx 0$.
Un $\lambda_1$ adapté est donc un antécédent de $-\nabla_x e(x_1)$ par $c'(x_1)^T$. En effet s'il existe on aura :

$$c'(x_1)^T \lambda_1 = -\nabla_x e(x_1)$$
$$\implies \nabla_x e(x_1) + c'(x_1)^T \times \lambda_1 = 0$$
$$\implies ||\nabla_x l(x_1, \lambda_1)||_\infty = 0 < \text{options.tol(1)}$$

On va donc utiliser la méthode matlab suivante :

```
lm = -a'\g ; %a est la jacobienne des contraintes et g le gradient de e
```

Cette méthode va résoudre un problème des moindres carrés à partir des données a' et g, elle est intéressante car peu coûteuse en calcul. De plus, elle nous fournit effectivement un antécédent de $-\nabla_x e(x_1)$ par $c'(x_1)^T$.

# 3 Globalisation par recherche linéaire

Dans cette dernière partie, nous allons améliorer la méthode de Newton introduite dans la partie précédente en globalisant la recherche de solution. Pour cela, nous utiliserons la recherche linéaire associée à la règle d'Armijo simplifiée.

## 3.1 Implémentation de la recherche linéaire

```
%%Recherche du pas alpha_k

% Initialisation de i_k,alpha_k et omega (w)
i_k = 0 ;
alpha_k = (1/2)^(i_k) ;
w = 10^(-4) ;

%Calcul de phi(z_k + alpha pk)
[~,c_k,g_k,a_k,~,~]=simul(4,x_k + alpha_k*d_k,lambda_k + alpha_k*mu_k);

F = [g_k+a_k'*(lambda_k + alpha_k*mu_k);c_k] ;

phi = (1/2)*F'*F ;

%Calcul de phi(z_k) + w alpha_k *phi'(z_k).pk
pk = [d_k ; mu_k] ;

phi_tot = (1/2)*(Fk'*Fk) + (w*alpha_k)*(Fk')*Fk_prime * pk ;

while phi > phi_tot
    i_k = i_k+1 ;
    alpha_k = alpha_k/2 ;
  %Calcul de phi(z_k + alpha pk)
  [~,c_k,g_k,a_k,~,~]=simul(4,x_k + alpha_k*d_k,lambda_k + alpha_k*mu_k);

  F = [+g_k+a_k'*lm;c_k] ;
  phi = (1/2)*F'*F ;

 %Calcul de phi(z_k) + w alpha_k *phi'(z_k).pk
  pk = [d_k ; mu_k] ;
  phi_tot = (1/2)*(Fk'*Fk) + (w*alpha_k)*(Fk')*Fk_prime * pk ;
end
  x = x_k + alpha_k*d_k ;
  lm = lambda_k + alpha_k*mu_k ;
```

Figure 26 : Implémentation de la recherche linéaire

La recherche linéaire se fait après avoir déterminé la direction $p_k$, on va chercher un pas $\alpha_k$ compris entre 0 et 1 qui va donner l'avancement de notre $z_k$ dans la direction $p_k$. On a alors $z_{k+1} = z_k + \alpha_k p_k$.
On choisit $\alpha_k$ comme une puissance de deux qui va vérifier l'inégalité :

$$\phi(z_k + \alpha_k p_k) \leq \phi(z_k) + \omega.\alpha_k \phi'(z_k).p_k$$

## 3.2 Propriétés de la recherche linéaire (troisième partie-TP3)

- Montrons que la direction $p_k$ donnée par l'algorithme de Newton est une direction de descente en $z_k$ pour la fonction $\phi$.
  Dans la méthode de recherche linéaire on définit $\phi$ par :

$$\phi(z) = \frac{1}{2}||F(z)||_2^2$$

  On définit dans la méthode de Newton F par :

$$F(z) = \begin{pmatrix} \nabla e(x) + c'(x)^T \lambda \\ c(x) \end{pmatrix} \text{ et } z = (x, \lambda) \in \mathbb{R}^N$$

  Sa différentielle vaut :

$$dF(z) = F'(z) = \begin{pmatrix} \nabla^2_{xx} l(x_k, \lambda_k) & c'(x)^T \\ c'(x_k) & 0 \end{pmatrix} \text{ et } z = (x, \lambda) \in \mathbb{R}^N$$

Or on définit $p_k$ par $p_k = (d_k, \mu_k)$ vérifiant

$$F'(z_k)\begin{pmatrix} d_k \\ \mu_k \end{pmatrix} = -\begin{pmatrix} \nabla e(x_k) + c'(x_k)^T \lambda_k \\ c(x_k) \end{pmatrix} = -F(z_k)$$

On peut réécrire $\phi(z)$ comme suit et en déduire l'expression de son gradient :

$$\phi(z) = \frac{1}{2}\langle F(z), F(z)\rangle$$

$$\implies \nabla\phi(z) = F(z)^T dF(z)$$

Ensuite, en réutilisant la définition de $p_k$ on a pour $t$ réel assez petit :

$$\frac{\phi(z_k + tp_k) - \phi(z_k)}{t} \simeq F(z_k)^T dF(z_k) p_k$$

$$\frac{\phi(z_k + tp_k) - \phi(z_k)}{t} \simeq F(z_k)^T (-F(z_k))$$

$$\frac{\phi(z_k + tp_k) - \phi(z_k)}{t} \simeq -||F(z_k)||_2^2 < 0 \quad \text{(en un point } z_k \text{ qui n'est pas solution)}$$

On peut alors conclure que $p_k$ est bien une direction de descente en $z_k$ pour $\phi$

- Expliquons pourquoi on essaye d'abord le pas $\alpha_k = 1$.
  Le fait de prendre le pas $\alpha_k = 1$ permet de ne pas avoir un pas $\alpha_k$ trop petit, dans le cas ou ce pas ne convient pas car il ne vérifie pas l'inégalité de la règle d'Armijo on peut alors le réduire en le divisant par 2. Le fait de ne pas prendre un pas de départ trop faible est essentiel, en effet si l'on prenait un pas trop faible on pourrait alors obtenir une succession de positions formant une suite de Cauchy (comme cela est décrit dans le cours) qui converge vers un point x localement qui n'est pas solution de notre problème. Plus généralement, un pas trop petit peut tout simplement empêcher la convergence de l'algorithme, c'est pourquoi nous prenons 1 pour pas de départ.

  Montrons qu'on est sûr de trouver un pas $\alpha_k$ vérifiant l'inégalité suivante

  $$\phi(z_k + \alpha_k p_k) \leq \phi(z_k) + \omega.\alpha_k \phi'(z_k).p_k$$

  Si $F(z_k) = 0$ alors on n'a pas besoin d'appliquer la règle d'Armijo : on a testé l'optimalité plus tôt dans l'algorithme et il s'est terminé.
  Si $F(z_k) \neq 0$ alors on a

  $$\lim_{t\to 0} \frac{\phi(z_k + tp_k) - \phi(z_k)}{t} = -||F(z_k)||_2^2$$

  Pour t suffisamment petit (par exemple $t \leq t_0$ avec $t_0$ qui dépend de $z_k$ et de $\omega \in [0, 1]$ ) on a :

  $$\frac{\phi(z_k + tp_k) - \phi(z_k)}{t} \leq -\omega||F(z_k)||_2^2$$

  car:

  $$-||F(z_k)||_2^2 \leq -\omega||F(z_k)||_2^2$$

  (et à partir d'un certain t petit $\dfrac{\phi(z_k + tp_k) - \phi(z_k)}{t}$ sera dans une boule centrée en $-||F(z_k)||_2^2$ et n'incluant pas $-\omega||F(z_k)||_2^2$, par convergence).

  On aura alors pour $\alpha_k < t_0$ :

  $$\phi(z_k + \alpha_k p_k) \leq \phi(z_k) + \alpha_k.\omega.\phi'(z_k) p_k$$

Passons désormais à la pratique en exécutant notre programme sur les cas-tests proposés.

### 3.3 Cas-tests pour l'algorithme de Newton avec recherche linéaire

- Cas test 2d: On a déjà introduit ce cas-test à la section précédente.
  La tolérance (options) sera prise à $10^{-3}$ pour le calcul d'une solution et on prendra un nombre maximal d'itérations égal à 1000.
  On obtient la position optimisée de la chaîne suivante:

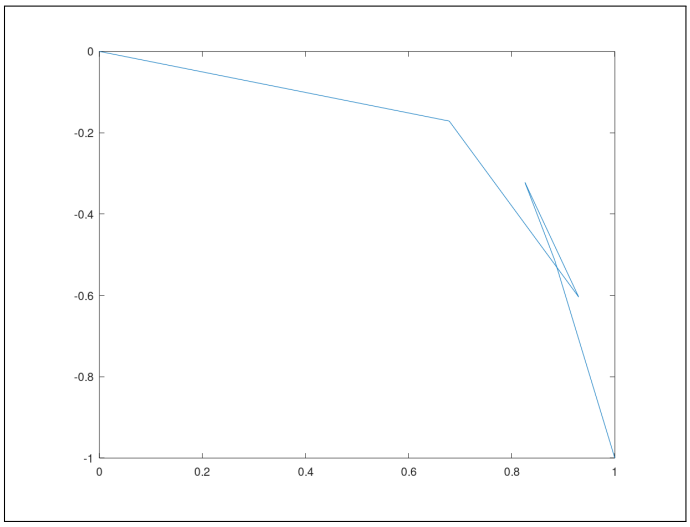



Figure 27 : Position optimisée de la chaîne (cas test 2d)

Pour plus de précision, le programme nous renvoie les vecteurs de positions des noeuds et des multiplicateurs lagrangiens (xy et lm) suivants:

```
x =

   0.67873
   0.92981
   0.82568
   0.88345
  -0.17126
  -0.60365
  -0.32230
  -0.51377

lm =

  -0.30072
  -0.81293
   1.96020
  -3.53311
  -1.75125
```

Par ailleurs, voici la sortie de l'optimiseur (avec options.verb = 1, sinon on peut obtenir plus d'informations avec options.verb=2):

```
-------------------------------------------------------------------------------

iter      |gl|          |ce|         |x|        |lm|        alpha         phi
    1   2.4956e-01   7.7026e+00   1.5e+00   6.4e-01   3.125e-02   6.57603e+01
    2   2.9238e-01   2.0650e+00   9.7e-01   1.1e+00   1.000e+00   5.60225e+00
    3   4.4940e-01   1.5837e+00   7.4e-01   1.3e+00   5.000e-01   3.01134e+00
    4   5.5588e-01   1.6550e+00   8.6e-01   1.8e+00   5.000e-01   2.66321e+00
    5   3.2658e-01   1.6276e+00   7.2e-01   2.5e+00   2.500e-01   2.17794e+00
    6   3.4954e-01   1.5776e+00   7.5e-01   6.1e-01   3.125e-02   2.12390e+00
    7   7.6374e-01   1.4987e+00   1.1e+00   1.1e+00   2.500e-01   2.08650e+00
    8   6.7127e-01   8.8351e-01   1.3e+00   6.4e-01   1.000e+00   9.24207e-01
    9   6.3062e-01   3.8424e-01   8.5e-01   1.3e+00   1.000e+00   4.41996e-01
   10   2.1263e-01   1.4549e-01   9.6e-01   1.5e+00   1.000e+00   5.25454e-02
   11   8.1372e-02   8.1474e-02   9.4e-01   1.7e+00   5.000e-01   1.51907e-02
   12   8.7556e-02   4.4406e-02   9.3e-01   2.1e+00   5.000e-01   9.04604e-03
   13   7.5649e-02   2.3725e-02   9.3e-01   2.5e+00   5.000e-01   5.76332e-03
   14   6.9040e-02   3.4034e-03   9.3e-01   3.2e+00   1.000e+00   4.55810e-03
   15   3.1844e-03   8.4017e-05   9.3e-01   3.5e+00   1.000e+00   1.13268e-05
   16   2.0759e-06   2.8775e-08   9.3e-01   3.5e+00   1.000e+00   4.86858e-12

-------------------------------------------------------------------------------
```

On s'aperçoit grâce au seconde et troisième colonne que le gradient du lagrangien et les contraintes tendent rapidement vers 0 en norme, ce qui est un signe favorable de convergence.
En effet, on a convergence pour moins de 20 itérations (16 dans notre cas) contrairement au résultat lors de l'utilisation de l'algorithme de Newton sans recherche linéaire:

info

| | Values |
|---|---|
| niter | 16 |
| status | 0 |

Pour connaître la nature du point de convergence, on calcule une base de vecteur propre de $\mathrm{Ker}(c')$, puis l'on concatène les différents vecteurs propres dans une matrice $K$, enfin l'on calcule les valeurs propres de la matrice $K^T LK$, où $L = \nabla^2_{xx} l(x_*, \lambda_*)$ ce qui nous permet de conclure sur la nature du point de convergence (cf annexe 1)):

```
[~,~,~,~,L,~]=chs(5,x,lm);
K=null(a);
[vecteurs_propres,valeurs_propres]=eig((K')*L*K);
valeurs_propres
```

Ici, il y a convergence de l'algorithme vers un **point selle** car les valeurs propres de la matrice $K^T LK$ sont strictement négatives et strictement positives:

```
valeurs_propres =

Diagonal Matrix

    4.9446         0         0
         0   -1.8333         0
         0         0  -15.0152
```

Passons désormais au cas-test suivant:

- Cas test 3a : nous avons ici 2 barres de longueur 0.6 et 0.6, de plus le second point de fixation de la chaîne est (a,b) = (1,0) et la position initiale du noeud est en (0.5,0.4)
  On obtient la représentation suivante de la chaîne initiale:

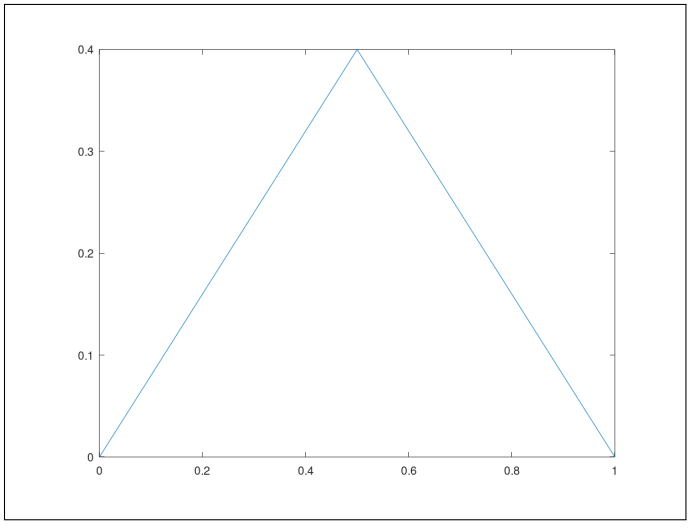



Figure 28 : Position initiale de la chaîne (cas test 3a)

La position de la chaîne renvoyée par l'optimiseur est la suivante:

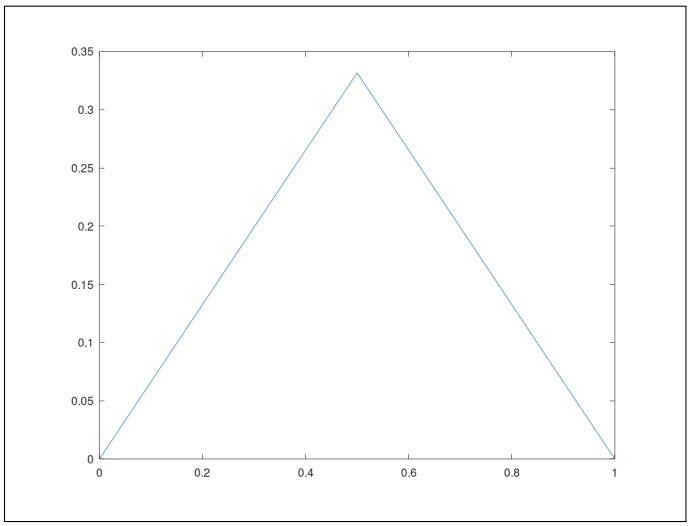



Figure 29 : Position optimisée de la chaîne (cas test 3a)

Pour plus de précision, le programme nous renvoie les vecteurs de positions des noeuds et des multiplicateurs lagrangiens (xy et lm) suivants:

```
x =          lm =

    0.50000   -0.45188
    0.33171   -0.45188
```

Par ailleurs, voici la sortie de l'optimiseur (avec options.verb = 1):

```
------------------------------------------------------------------------------------

iter       |gl|           |ce|          |x|         |lm|        alpha         phi
    1    1.4648e-02    3.9062e-03    5.0e-01    4.3e-01    1.000e+00    1.22547e-04
    2    4.2327e-04    3.3490e-05    5.0e-01    4.5e-01    1.000e+00    9.07019e-08

------------------------------------------------------------------------------------
```

On s'aperçoit grâce au seconde et troisième colonne que le gradient du lagrangien et les contraintes tendent rapidement vers 0 en norme, ce qui est un signe favorable de convergence.
Il y a convergence vers la solution en 2 itérations:

info

| | Values |
|---|---|
| niter | 2 |
| status | 0 |

On a convergence ici vers un maximum, en effet il faut prendre en compte que dans un problème avec deux barres de longueurs fixées tel que (a,b) $\neq$ (0,0), on a alors l'ensemble admissible qui est de cardinal au plus 2 (intersection de deux cercles de rayon respectif la longueur des barres et de centre (0,0) et (a,b)). Aussi, si l'on atteint l'un des points admissibles il est alors très simple de déterminer sa nature. Par exemple, ici les deux positions admissibles sont les suivantes:

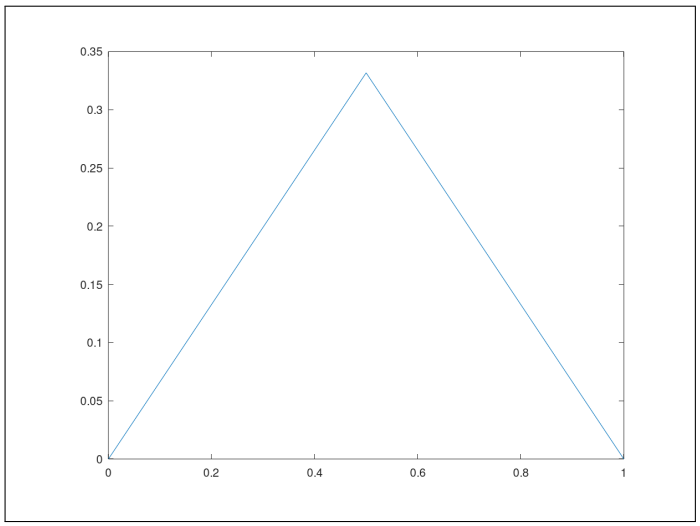
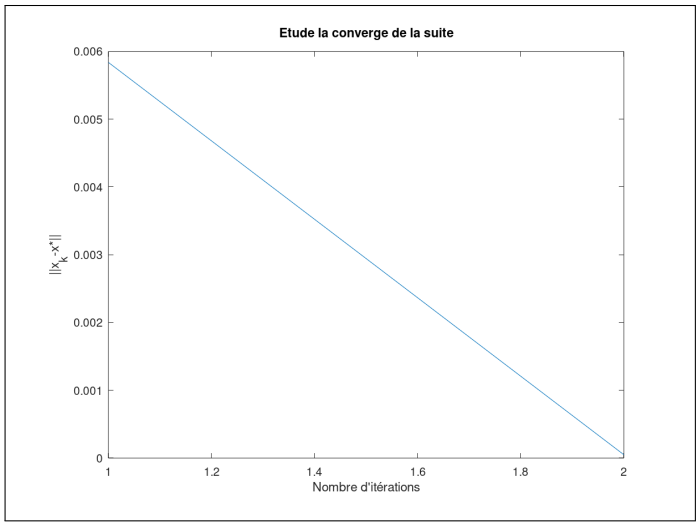



Figure 30 : Position admissible d'énergie maximale de la chaîne (cas test 3a)

Cette position correspond à un noeud de position $(0.5, \sqrt{0.11})$ (vérifie les contraintes d'après le théorème de Pythagore), l'énergie du système est alors $\boxed{e \simeq 0.199 \text{ J}}$, de plus **l'algorithme nous fait converger vers cette position**. La figure obtenue et les résultats numériques nous permettent de déduire cela, par ailleurs pour un résultat plus formel on peut tracer l'évolution de $||x_k - x_*||$ à chaque itération, où $x_* = (0.5, \sqrt{0.11})$




Ce qui permet de confirmer la convergence vers cette position.
Par ailleurs la seconde position admissible est:

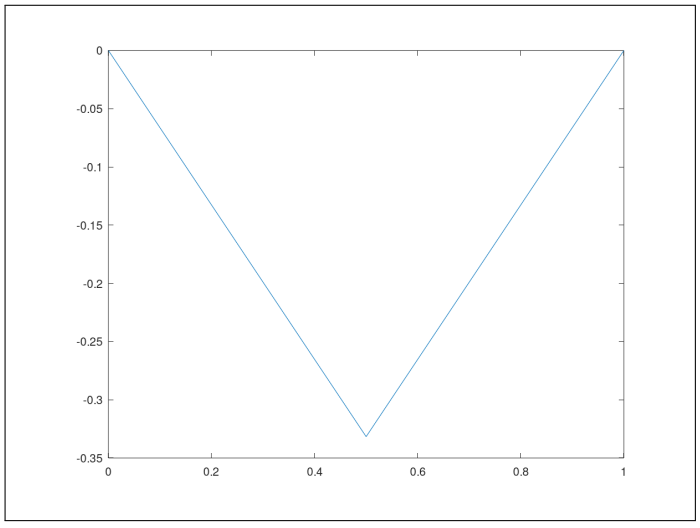



Figure 31 : Position admissible d'énergie minimale de la chaîne (cas test 3a)

Cette position correspond à un noeud de position $(0.5,-\sqrt{0.11})$ (vérifie les contraintes d'après le théorème de Pythagore), l'énergie du système est alors $\boxed{e \simeq -0.199 \text{ J}}$.
Ainsi, on peut conclure que l'algorithme de Newton avec recherche linéaire par la règle d'Armijo nous fait converger vers un maximum (en effet la première position correspond à un maximum et la seconde à un minimum d'après le calcul de l'énergie avec le simulateur).
Passons désormais au cas-test suivant:

- Cas test 3b : Nous avons ici 2 barres de longueur respective 2 et 1, de plus le second point de fixation de la chaîne est (a,b) = (1,0) et la position initiale du noeud est en (0.5,0.3). Nous obtenons la représentation initiale suivante de la chaîne:

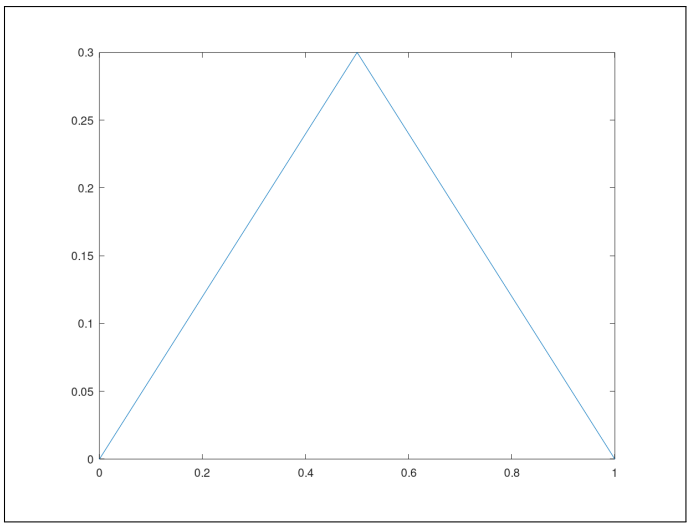



Figure 32 : Position initiale de la chaîne (cas test 3b)

La position renvoyée par le programme est la suivante:

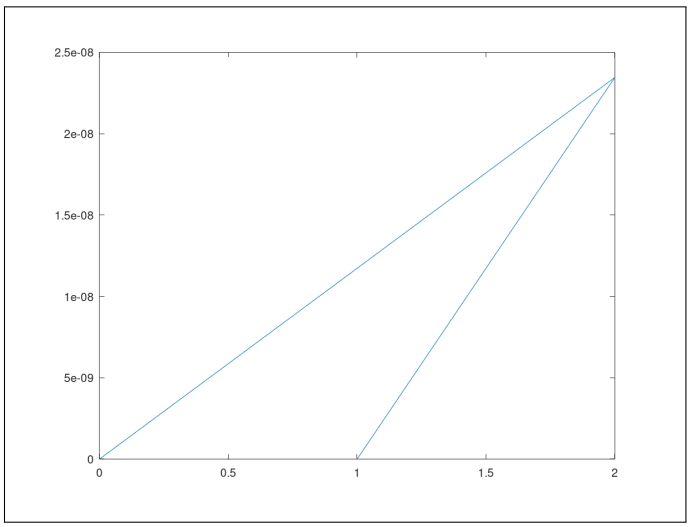



Figure 33 : Position optimisée de la chaîne (cas test 3b)

On peut remarquer que l'ordonnée est très faible sur la figure précédente.
Pour plus de précision, le programme nous renvoie les vecteurs de positions des noeuds et des multiplicateurs lagrangiens (xy et lm) suivants:

```
x =                          lm =

   2.000000000000            31967062.72211
   0.000000023462           -63934125.44422
```

Par ailleurs, voici un extrait de la sortie de l'optimiseur (avec options.verb = 1):

```
------------------------------------------------------------------------------

iter       |gl|           |ce|          |x|         |lm|         alpha          phi
    1    8.4375e-01    3.3718e+00    5.9e-01    5.5e-01    6.250e-02    6.25882e+00
    2    1.3345e+00    8.7152e-01    1.3e+00    5.6e-01    5.000e-01    1.88118e+00
    3    7.3411e-01    7.2730e-01    2.0e+00    8.0e-01    1.000e+00    9.25374e-01
    4    5.0097e-01    1.8183e-01    2.0e+00    2.3e+00    1.000e+00    1.58546e-01
    5    5.0024e-01    4.5456e-02    2.0e+00    4.7e+00    1.000e+00    1.27188e-01
    6    5.0006e-01    1.1364e-02    2.0e+00    9.4e+00    1.000e+00    1.25159e-01
    7    5.0002e-01    2.8410e-03    2.0e+00    1.9e+01    1.000e+00    1.25016e-01
    8    3.7501e-01    1.5981e-03    2.0e+00    2.8e+01    5.000e-01    7.03183e-02
    9    3.0469e-01    8.9892e-04    2.0e+00    4.0e+01    5.000e-01    4.64195e-02
   10    2.6514e-01    5.0564e-04    2.0e+00    5.5e+01    5.000e-01    3.51497e-02
   11    2.4289e-01    2.8442e-04    2.0e+00    7.5e+01    5.000e-01    2.94981e-02
   12    2.3038e-01    1.5999e-04    2.0e+00    1.0e+02    5.000e-01    2.65366e-02
   13    2.2334e-01    8.9993e-05    2.0e+00    1.3e+02    5.000e-01    2.49396e-02
   14    2.1938e-01    5.0621e-05    2.0e+00    1.8e+02    5.000e-01    2.40631e-02
   15    2.1715e-01    2.8474e-05    2.0e+00    2.4e+02    5.000e-01    2.35769e-02
   16    2.1590e-01    1.6017e-05    2.0e+00    3.2e+02    5.000e-01    2.33057e-02
   17    2.1519e-01    9.0095e-06    2.0e+00    4.3e+02    5.000e-01    2.31538e-02
   18    2.1480e-01    5.0678e-06    2.0e+00    5.7e+02    5.000e-01    2.30685e-02
   19    2.1457e-01    2.8507e-06    2.0e+00    7.6e+02    5.000e-01    2.30207e-02
   20    2.1445e-01    1.6035e-06    2.0e+00    1.0e+03    5.000e-01    2.29938e-02
```

On s'aperçoit grâce au seconde et troisième colonne que le gradient du lagrangien et les contraintes tendent vers 0 en norme, toutefois cette convergence est moins rapide que dans les autres cas-tests. D'ailleurs, on peut aussi remarquer que la méthode de Newton sans recherche linéaire est ici plus rapide que la méthode avec recherche linéaire.
En effet, il y a convergence vers la solution en 195 itérations ce qui est élevé en comparaison avec les autres cas-tests (31 itérations suffisent si l'on utilise la méthode de Newton sans recherche linéaire, aussi ceci met en valeur le fait que la méthode de Newton avec recherche linéaire peut avoir pour avantage de trouver des solutions là où la méthode de Newton seul ne le pourrait pas (cf cas 2d), toutefois cette méthode avec recherche linéaire peut être plus lente pour des cas simples).

info

| | Values |
|---|---|
| niter | 195 |
| status | 0 |

En reprenant le raisonnement du cas-test précédent on peut déduire de la nature de la position de convergence: il faut prendre en compte que dans un problème avec deux barres de longueurs fixées tel que (a,b) $\neq$ (0,0), on a alors l'ensemble admissible qui est de cardinal au plus 2 (intersection de deux cercles de rayon respectif la longueur des barres et de centre (0,0) et (a,b)). Aussi, si l'on atteint l'un des points admissibles il est alors très simple de déterminer sa nature.
Par exemple, ici il n'y qu'une seule position admissible qui est la suivante:

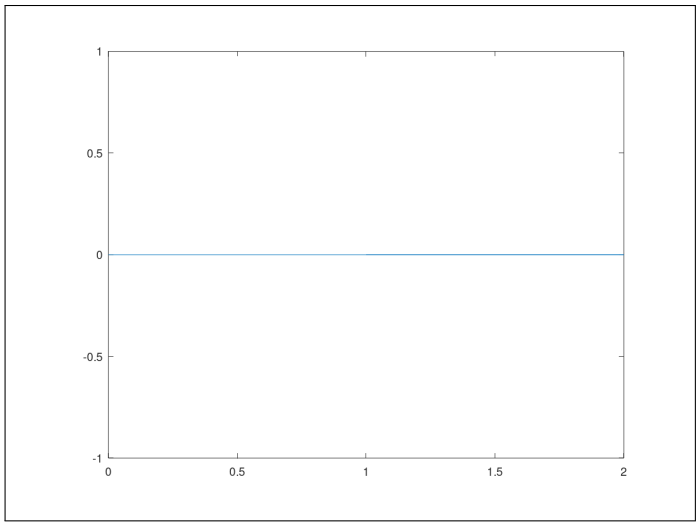
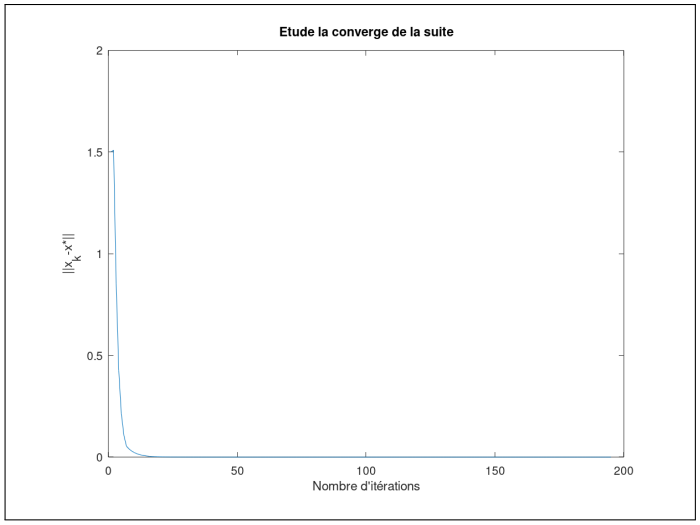



Figure 34 : Unique position admissible de la chaîne (cas test 3b)

Cette position correspond à un noeud de position (2,0), l'énergie du système est alors $\boxed{e \simeq 0 \text{ J}}$, de plus **l'algorithme nous fait converger vers cette position**. La figure obtenue et les résultats numériques nous permettent de déduire cela, par ailleurs pour un résultat plus formel on peut tracer l'évolution de $||x_k - x_*||$ à chaque itération, où $x_* = (2, 0)$




Ce qui permet de confirmer la convergence vers cette position.
Aussi, cette position correspondant à l'unique position admissible est donc la solution du problème (c'est un minimum étant donné que l'ensemble admissible est réduit à un singleton). Ainsi, on peut conclure que l'algorithme de Newton avec recherche linéaire par la règle d'Armijo nous fait converger vers l'unique solution du problème.
Passons désormais au dernier cas-test:

- Cas test 3c : Nous avons désormais 2 barres de longueur respectives 2 et 1, de plus le second point de fixation de la chaîne est (a,b) = (0,-1) et la position initiale du noeud est en (0.3,0.3).
On obtient la représentation initiale suivante:

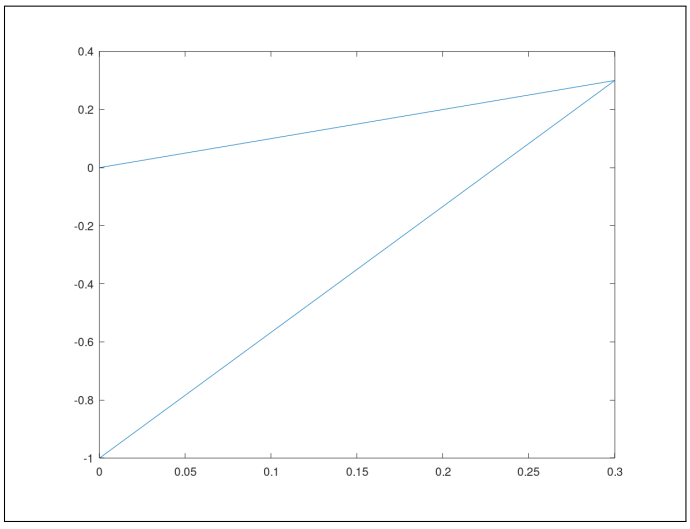



Figure 35 : Position initiale de la chaîne (cas test 3c)

Enfin, le programme nous renvoie la position optimisée suivante:

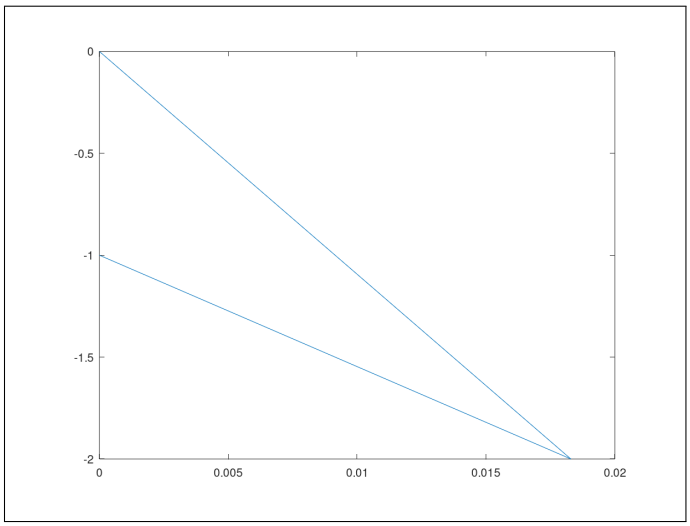



Figure 36: Position optimisée de la chaîne (cas test 3c)

C'est désormais l'abscisse qui est très faible sur la figure précédente, aussi le noeud est approximativement située en (-2,0).
Pour plus de précision, le programme nous renvoie les vecteurs de positions des noeuds et des multiplicateurs lagrangiens (xy et lm) suivants:

```
x =           lm =

   0.018294     0.75000
  -2.000000    -0.75000
```

Par ailleurs, voici la sortie de l'optimiseur (avec options.verb = 1):

```
---------------------------------------------------------------------------------

iter       |gl|          |ce|         |x|        |lm|        alpha         phi
    1   2.2204e-16   2.0862e+00   1.4e+00   7.5e-01   1.250e-01   4.05559e+00
    2   0.0000e+00   2.1310e+00   1.8e+00   7.5e-01   5.000e-01   2.27763e+00
    3   0.0000e+00   1.3709e+00   2.0e+00   7.5e-01   1.000e+00   1.87930e+00
    4   0.0000e+00   3.4272e-01   2.0e+00   7.5e-01   1.000e+00   1.17456e-01
    5   0.0000e+00   8.5680e-02   2.0e+00   7.5e-01   1.000e+00   7.34101e-03
    6   0.0000e+00   2.1420e-02   2.0e+00   7.5e-01   1.000e+00   4.58813e-04
    7   0.0000e+00   5.3550e-03   2.0e+00   7.5e-01   1.000e+00   2.86758e-05
    8   0.0000e+00   1.3387e-03   2.0e+00   7.5e-01   1.000e+00   1.79224e-06
    9   0.0000e+00   3.3469e-04   2.0e+00   7.5e-01   1.000e+00   1.12015e-07

---------------------------------------------------------------------------------
```

On s'aperçoit grâce au seconde et troisième colonne que le gradient du lagrangien et les contraintes tendent rapidement vers 0 en norme, ce qui est un signe favorable de convergence.
Il y a convergence vers la solution en 9 itérations.

| info | |
|---|---|
| | Values |
| niter | 9 |
| status | 0 |

En reprenant le raisonnement du cas-test précédent on peut déduire de la nature de la position de convergence: il faut prendre en compte que dans un problème avec deux barres de longueurs fixées tel que (a,b) $\neq$ (0,0), on a alors l'ensemble admissible qui est de cardinal au plus 2 (intersection de deux cercles de rayon respectif la longueur des barres et de centre (0,0) et (a,b)). Aussi, si l'on atteint l'un des points admissibles il est alors très simple de déterminer sa nature. Par exemple, ici il n'y qu'une seule position admissible qui est la suivante:

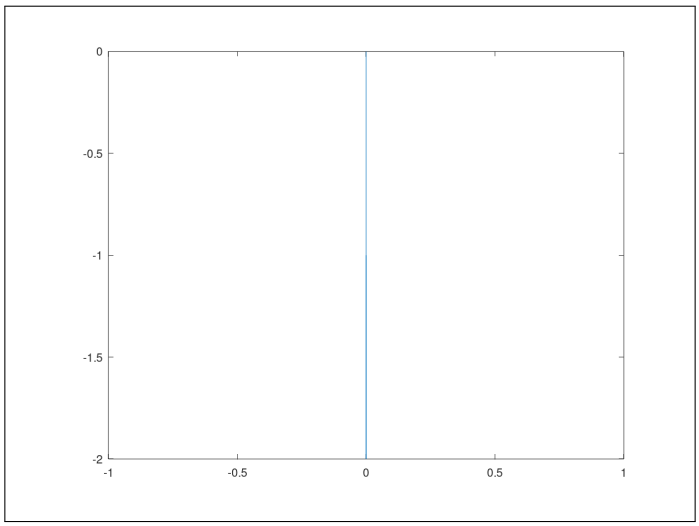
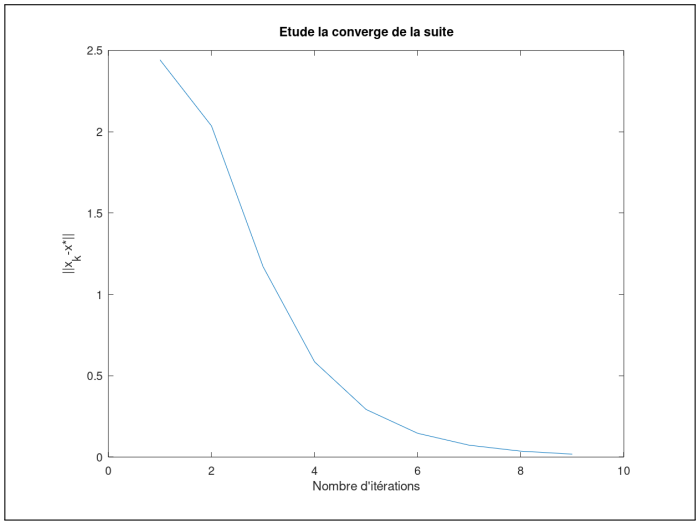



Figure 37 : Unique position admissible de la chaîne (cas test 3c)

Cette position correspond à un noeud de position (0,-2), l'énergie du système est alors $\boxed{e \simeq -3.5 \text{ J}}$, de plus **l'algorithme nous fait converger vers cette position**. La figure obtenue et les résultats numériques nous permettent de déduire cela, par ailleurs pour un résultat plus formel on peut tracer l'évolution de $||x_k - x_*||$ à chaque itération, où $x_* = (0, -2)$:




Ce qui permet de confirmer la convergence vers cette position.
Aussi, cette position correspondant à l'unique position admissible est donc la solution du problème (c'est un minimum étant donné que l'ensemble admissible est réduit à un singleton). Ainsi, on peut conclure que l'algorithme de Newton avec recherche linéaire par la règle d'Armijo nous fait converger vers l'unique solution du problème.

# Conclusion

Pour conclure, nous avons exposé la modélisation mathématique de notre problème, afin de le mettre sous la forme d'un problème de minimisation sous contraintes d'égalités. Par la suite, nous avons traité l'implémentation d'un simulateur piloté par l'utilisateur permettant d'obtenir différents résultats clés de notre problème. Nous avons mis en place l'optimiseur qui va nous permettre de résoudre différents problèmes d'optimisation (on se restreindra toutefois à l'équilibre d'une chaine articulée). Pour cela, nous avons implémenté la méthode de Newton qui calculait les données intéressantes à chaque itération grâce au simulateur. Enfin nous avons étudié la convergence de la solution et nous avons réalisé une routine nous permettant de savoir si la solution était un minimum, un maximum ou un point-selle. On a alors été confronté à des problèmes de convergence avec certains cas test qu'on n'arrivait pas à résoudre car ils ne convergeaient pas lorsque l'on appliquait l'algorithme de Newton. On a alors mis en place la méthode de recherche linéaire qui nous permet de définir un pas d'avancement dans la direction de descente à chaque itération, on a pu alors obtenir une convergence pour le cas test 2d. De plus, nous avons finalement éprouvé le fonctionnement de la méthode de Newton associé à la recherche linéaire sur des cas-test singuliers à deux barres. Ainsi, ce projet nous a permis de découvrir la modélisation mathématique et informatique d'un problème d'optimisation en utilisant la méthode de Newton associé notamment à la recherche linéaire utilisant la règle d'Armijo pour globaliser la recherche de solution.

# Annexe

## 1)

On va démontrer dans cette partie comment la routine implémentée nous permet de conclure sur la nature de la solution (minimum, maximum, point-selle).

Soit $z^*$ notre point stationnaire : on a $||c(x_1)||_\infty = 0$ et $||\nabla_x l(x^*, \lambda^*)||_\infty = 0$. On a par ailleurs que $e$ et $c$ sont dérivables. Or d'après le cours sous ces hypothèses, une condition suffisante pour que $x^*$ soit un minimum local stricte et que l'on ait :

$$\forall d \in T_{x_*} X_E \backslash \{0\} : \langle L_* d, d \rangle > 0, \text{ où L est la hessienne du lagrangien}$$

Alors $x*$ est un minimum local stricte de $(P_E)$.

On sait que les directions qui nous permettent de "rester" dans l'espace des contraintes sont celles qui vérifient $c'(x_*).d = 0$, en clair ça correspond au noyau de $c'(x_*)$ qui est la jacobienne des contraintes.

Soit $B = (e_1, ..., e_p)$ une base de $\text{Ker}(c(x_*))$, on construit $A$ une matrice dont les colonnes sont les vecteurs de la base B et on a

$$\forall d \in T_{x_*} X_E, \exists v \in \mathbb{R}^n : d = Av$$

Ainsi la condition suffisante énoncée plus haut devient :

$$\forall v \in \mathbb{R}^n \backslash \{0\} : \langle L_* Av, Av \rangle > 0$$

$$\implies \forall v \in \mathbb{R}^n \backslash \{0\} : \langle A^T L_* Av, v \rangle > 0$$

$$\implies A^T \times L \times A \text{ est définie positive}$$

Si on a ce résultat on peut directement conclure que $x_*$ est un minimum local stricte.

Une condition nécessaire pour que le point soit un minimum local (pas forcément strict) est donnée par :

$$\forall d \in T_{x_*} X_E \backslash \{0\} : \langle L_* d, d \rangle \geq 0, \text{ où L est la hessienne du lagrangien}$$

On effectue le même raisonnement que précédemment et en on conclut que la matrice $M = A^T \times L \times A$ doit nécessairement être semi-définie positive pour qu'on ait un minimum local en $x*$ On peut calculer cette matrice sur matlab et récupérer ces valeurs propres ce qui nous permet de conclure sur la nature du point stationnaire, on a alors les possibilités suivantes :

- M est définie positive on a un minimum local strict en $x*$
- -M est définie positive et on a un maximum local strict en $x*$
- M a des valeurs propres positives et négatives et on a un point-selle en $x*$