Rapport des TP de OPT201 et OPT202 : Équilibre d'une chaîne articulée

Mathieu Petitbois & Hugo Blanc

5 mars 2021

Table des matières

1 Introduction

Le but de ce TP est de modéliser et résoudre un problème d'équilibre d'une chaîne constituée de n_b barres rigides et de $n_n = n_b - 1$ noeuds (sans compter les noeuds aux extrémités de la chaîne qui correspondent à des constantes du problème), cette chaîne se trouvant par ailleurs au-dessus d'un sol rigide. L'objet des travaux décrits dans ce rapport est l'établissement d'une méthode algorithmique permettant la détermination, à partir d'une condition initiale en position d'une chaîne, de la position "naturelle" qu'elle prendrait sous la contrainte de la gravité. Pour ce faire, nous procéderons par étape en nous appuyant sur le langage de programmation Matlab :

— Dans un premier temps, nous établirons une modélisation du problème simplifié dans le cas d'absence de sol afin d'écrire un simulateur permettant de calculer de manière optimisée les grandeurs du problème. (TP1)

— Ensuite, nous implémenterons un premier algorithme dit de Newton permettant la résolution partielle du problème simplifié (trouver des points stationnaires). (TP2)

— Dans un troisième temps, nous améliorerons notre premier algorithme de Newton dans la cas simplifié d'absence de sol en implémentant des techniques dites de globalisation élargissant ses domaines de convergence. (TP3)

— Dans un quatrième temps, nous considérerons la présence d'un sol et implémenterons un nouvel algorithme de Newton dit OQS adapté au nouveau problème. (TP4)

— Enfin, nous améliorerons le précédent algorithme en lui introduisant une version quasi-newtonienne : l'algorithme BFGS. (TP5)

2 Écriture du modèle simplifié et établissement du simulateur : `chs.m` (TP1)

2.1 Modèle du cas simplifié

Sans considérer la présence d'un sol, notre problème (représenté ci-dessous) peut se ramener à un problème d'optimisation.

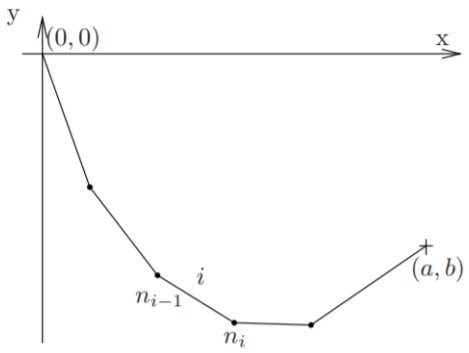


FIGURE 1 – Représentation de la chaîne

Les principes de la physique nous enseignent en effet que la position prise par la chaîne sera cella qui minimise son énergie potentielle gravitationnelle. En somme, on peut montrer que le problème se ramène à la minimisation suivante (les contraintes d'égalités étant dues à la rigidité des barres) :

$$(P_E) : \begin{cases} \min\limits_{xy \in \mathbb{R}^n} e(xy) \\ c_i(xy) = 0, \forall i \in E \end{cases} \Leftrightarrow \min_{xy \in XY} e(xy)$$

avec :

- $n = 2 \times n_n$ le nombre paramètres variables de notre problème.
- $xy = (x_1, ..., x_{n_n}, x_{n_n+1}, y_0, y_1, ..., y_{n_n})$ le vecteur des paramètres auquel on pourra ajouter plus tard les coordonnées des extrémités (x_0, y_0) et (x_{n_n+1}, y_{n_n+1}) pour simplifier les calculs.
- $XY = \{xy \in \mathbb{R}^n, c_i(xy) = 0 \ \forall i \in E\}$ l'ensemble réalisable de notre problème P.
- $E = \{1, ..., n_b\}$ l'ensemble des indices des barres rigides.

— $\forall i \in E, c_i(xy) = (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 - L_i$ la fonction de contrainte sur la barre d'indice i.

— L_i la longueur de la barre d'indice i.

— $e(xy) = \sum_{i=1}^{n_b} L_i \frac{y_i + y_{i-1}}{2}$ l'énergie potentielle simplifiée dans le cadre de notre ensemble réalisable.

2.2 Existence d'une solution

Avant d'aller plus loin, il semble pertinent de déterminer mathématiquement s'il existe une solution à (P_E), ce qui est un prérecquis à la convergence de nos futurs algorithmes. On pourra supposer l'existence d'un point admissible, ce qui revient à dire que XY est non-vide, ce qui va de soi en supposant les longueurs L_i positives.

Or, on se trouve en dimension finie. On peut vérifier que XY est fermé. En effet, en posant $(xy_k)_{k\in\mathbb{N}}$ une suite d'éléments dans XY qui converge vers $xy \in \mathbb{R}^n$, on a :

$$\forall k \in \mathbb{N}, \forall i \in E, c_i(xy_k) = ((xy_k)_i - (xy_k)_{i-1})^2 - ((xy_k)_{n_n+i} - (xy_k)_{n_n+i-1})^2 - L_i = 0$$

Donc, en passant à la limite, par continuité des fonction c_i :

$$\forall i \in E, c_i(xy) = 0 \text{ donc } xy \in XY$$

On en déduit que XY est fermé. De plus, il est aisé de voir que XY est borné. En effet, XY correspond à l'ensemble des jeux de coordonnées des chaînes dont les longueurs des barres correspondent avec les longueurs $L_i, i \in E$. Ainsi, tous les noeuds de XY seront compris dans la boule de centre 0 et de rayon la somme des longueurs L_i. XY est donc borné. Étant en dimension finie, on déduit que XY est un compact.

De plus, e étant une fonction continue dans XY par somme de fonctions continues, on en déduit qu'elle atteint un minimum dans XY.

Il existe bien au moins une solution au problème P_E ainsi posé.

2.3 Existence d'un multiplicateur de lagrange optimal

On note xy_* une solution d'étude qui existe par ce qui précède. Le lagrangien du système est définit par :

$$l(xy, \lambda_x) = e(xy) + \lambda^{\mathbf{T}} c(xy)$$

On veut savoir s'il existe un multiplicateur de lagrange optimal λ_* tel que :

$$\nabla_x l(xy_*, \lambda_*) = \nabla e(xy_*) + \sum_i (\lambda_*)_i \nabla c_i(xy_*) = 0$$

Une condition nécéssaire (CN1 de Lagrange) pour qu'un tel multiplicateur existe est que :

— e et c doivent être différentiables en xy_*

— La contrainte c doit être qualifiée en xy_*

Or, e et $c_i, i = 1..n_b$ étant polynomiales définies sur $\mathbb{R}^n$, elles y sont différentiables et donc en particulier en xy_*.

On veut donc montrer que la contrainte c est qualifiée en xy_*. Une condition suffisante est la surjectivité de $c'(xy_*)^{\mathbf{T}}$. Or, $Im(c'(_*xy)^{\mathbf{T}} = ker(c'(xy_*))^{\perp}$ donc la surjectivité revient à dire que $Im(c'(xy_*))^{\mathbf{T}} = ker(c'(xy_*))^{\perp} = \mathbb{R}^{n_b}$ et donc que $ker(c'(xy_*)) = 0$. Ainsi, la qualification de la contrainte est vérifiée si :

$$\forall (\mu_i \geq 0)_{i \in \{1,...,n_b\}}, \sum_{i=1}^{n_b} \mu_i \nabla c_i(xy_*) = 0 \Rightarrow \forall i = 1, .., n_b, \mu_i = 0$$

Or, pour $(\mu_i \geq 0)_{i \in \{1,...,n_b\}}$ tel que :

$$\sum_{i=1}^{n_b} \mu_i \nabla c_i(xy_*) = 0$$

on a en calculant la jacobienne de c :

$$\forall i \in \{1, ..., n_n\}, \begin{cases} \mu_{i+1}(x_*^{i+1} - x_*^i) = \mu_i(x_*^i - x_*^{i-1}) \\ \mu_{i+1}(y_*^{i+1} - y_*^i) = \mu_i(y_*^i - y_*^{i-1}) \end{cases}$$

Par passage au carré et sommation, xy_* étant admissible,

$$\forall i \in \{1, ..., n_n\}, \mu_i^2 L_i^2 = \mu_{i+1}^2 L_{i+1}^2$$

et donc vu que les L_i sont non-nuls, tous les μ_i sont nuls si et seulement si un des μ_i est nul.

En supposant par l'absurde que les μ_i sont tous strictement positifs, on a :

$$\forall i \in \{1, ..., n_n\}, \frac{L_i}{L_{i+1}} = \frac{\mu_i}{\mu_{i+1}}$$

Ainsi, on trouve :

$$\forall i \in \{1, ..., n_n - 1\}, \begin{cases} (x_*^{i+2} - x_*^{i+1}) = \frac{L_{i+1}}{L_{i+2}}(x_*^{i+1} - x_*^i) \\ (y_*^{i+2} - y_*^{i+1}) = \frac{L_{i+1}}{L_{i+2}}(y_*^{i+1} - y_*^i) \end{cases}$$

Donc par récurrence :

$$\forall i \in \{1, ..., n_n\}, \begin{cases} (x_*^{i+1} - x_*^i) = \frac{L_1}{L_{i+1}}(x_*^1 - x_*^0) \\ (y_*^{i+1} - y_*^i) = \frac{L_1}{L_{i+1}}(y_*^1 - y_*^0) \end{cases}$$

Et sachant que les points extrêmes sont fixés : $(x_*^0, y_*^0) = (0, 0)$ et$(x_*^{n_n+1}, y_*^{n_n+1}) = (a, b)$, on en déduit de chaque coefficient $i \in \{0, ..., n_n + 1\}$:

$$x_*^i = \frac{\sum_{j=1}^{i} \frac{1}{L_j}}{\sum_{j=1}^{n_n+1} \frac{1}{L_j}} a \text{ et } y_*^i = \frac{\sum_{j=1}^{i} \frac{1}{L_j}}{\sum_{j=1}^{n_n+1} \frac{1}{L_j}} b$$

Ainsi, dans le cas où une solution de cette forme ne peut exister, on dira par l'absurde que c est bien qualifiée et qu'il existe un multiplicateur de Lagrange optimal.

Dans le cas contraire, on ne peut pas conclure. Cherchons donc un cas on il n'y a pas de multiplicateur de Lagrange optimal. Par ce qui précède, on sait déjà qu'il s'agira de chaînes sous forme de droites, où tous les noeuds sont alignés.

Par exemple, pour $(x_0, y_0) = (0, 0)$ et $(x_2, y_2) = (2, 0)$ avec $L = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, l'équation :

$$\nabla_x l(xy_*, \lambda_x) = e(xy_*) + \lambda^{\mathbf{T}} c(xy_*) = 0$$

devient :

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 2 & -2 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \lambda_*^1 \\ \lambda_*^2 \end{pmatrix} = 0$$

car alors $xy_* = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. On obtient alors bien une solution sous forme de droite et il n'existe cette fois pas de multiplicateur de Lagrange optimal.

Ainsi, sans le critère de surjectivité, l'existence d'un multiplicateur de Lagrange optimal pour notre problème n'est pas certaine.

2.4 Implémentation des scripts d'initialisation et du simulateur

Après ces considérations théoriques, nous pouvons débuter l'implémentation d'une méthode numérique de résolution de notre problème P_E. Nous nous concentrerons ici sur le script d'initialisation définissant les variables et le simulateur permettant le calcul des grandeurs pertinentes du problème. Nous utiliserons pour ce faire le langage Matlab particulièrement optimisé pour les calculs vectoriels dont on maximisera l'utilisation.

2.5 Le simulateur `chs.m`

Le simulateur est une fonction `chs` qui prend en argument :

— `indic` : le pilotage de l'action du simulateur.

— `xy` : le vecteur-colonne contenant d'abord les abscisses $\{x_i\}_{i=1}^{n_n}$ des nœuds, puis leurs ordonnées $\{y_i\}_{i=1}^{n_n}$.

— `lm` : le vecteur-colonne de dimension n_b contenant les multiplicateurs de Lagrange pour les contraintes d'égalité, $\lambda = \{\lambda_i\}_{i=1}^{n_b}$.

et renvoit en sortie :

— `e` : la valeur de l'énergie potentielle en `xy` (c'est-à-dire pour les nœuds dont les coordonnées sont dans `xy`).

— `c` : la valeur en `xy` des contraintes sur la longueur des barres, `c` étant un vecteur-colonne de dimension n_b.

— `g` : le gradient de e en `xy`, c'est un vecteur-colonne de dimension $2n_n$.

— `a` : la jacobienne des contraintes d'égalité en `xy`, une matrice de n_b lignes et $2n_n$ colonnes.

— `hl` : hessien du lagrangien en `xy` et `lm`.

— `indic` : l'indicateur des résultats de la simulation (0 dans le cas normal et 1 en cas d'erreurs sur les entrées).

Les données des longueurs `L` et des coordonnées de l'extrémité droite `(A,B)` seront communiquées en tant que variables globales issues de la fonction d'initialisation. L'extrémité gauche sera en l'origine du repère.

2.5.1 Calcul de `e`

Comme dit précédemment, il est beaucoup plus efficace d'utiliser le calcul vectoriel pour calculer les valeurs pertinentes. Pour ce faire, on doit tout d'abord

ajouter les coordonnées des points extrêmes `(0,0)` et `(A,B)` au vecteur `xy` puis séparer les abscisses et les ordonnées pour obtenir les vecteurs :

`x = [0;xy(1:nn);A]` et `y = [0;xy(nn+1:n);B]`

En posant ensuite la matrice M_p de taille n_b par $n_b + 1$ définie par :

$$M_p = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 1 \end{pmatrix}$$

et $y =^{\mathbf{T}} (y_0, ..., y_{n_b})$, on obtient bien :

$$\forall i \in \{1, ..., n_b\}, (M_p \times y)_i = \sum_{j=1}^{n_b+1} (M_p)_{i,j} \times y_j = y_i + y_{i-1}$$

Et ainsi avec $L =^{\mathbf{T}} (L_0, ..., L_{n_b})$, on a :

$$e = \frac{1}{2}^{\mathbf{T}} L \times M_p y = \sum_{i=1}^{n_b} L_i \frac{y_i + y_{i-1}}{2}$$

2.5.2 Calcul de c

Si on pose cette fois la matrice M_n de taille n_b par $n_b + 1$ définie par :

$$M_n = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix}$$

et $x =^{\mathbf{T}} (x_0, ..., x_{n_b})$ et $y =^{\mathbf{T}} (y_0, ..., y_{n_b})$, on a bien :

$$\forall i \in \{1, ..., n_b\}, (M_p \times x)_i = x_i - x_{i-1} \text{ et } (M_p \times y)_i = y_i - y_{i-1}$$

En appliquant l'opération de mise au carré γ coefficient par coefficient, il en résulte que :

$$\forall i \in \{1, ..., n_b\}, (\gamma M_n x + \gamma M_n y - \gamma L)_i = (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 - L_i^2 = c_i(xy)$$

Donc,

$$c = \gamma M_n x + \gamma M_n y - \gamma L$$

2.5.3 Calcul de g

On sait que g correspond au vecteur gradient de e en `xy`. Or, e ne dépendant pas des abscisses, on sait déjà que g aura ses n_n premiers termes nuls. Pour les n_n derniers termes en y, on sait que :

$$\forall i \in \{1, ..., n_n\}, \frac{\partial e}{\partial y_i}(xy) = \frac{L_i}{2} + \frac{L_{i+1}}{2}$$

car chaque $y_i, i \in \{1, ..., n_n\}$ intervient deux fois dans la somme. On en déduit alors la forme du vecteur g :

$$g = (\underbrace{0, ..., 0}_{n_n \text{ fois}}, \frac{L_1 + L_2}{2}, ..., \frac{L_{n_n} + L_{n_n+1}}{2})$$

2.5.4 Calcul de a

Pour construire théoriquement `a`, on distingue les cas analogues pour les asbcisses et les ordonnées et on procède par un raisonnement ligne i par ligne i pour i allant de 1 à n_b. Ainsi, la i^{me} ligne de la jacobienne de c pour $(x_1, ...x_{n_n})$ est le vecteur dont les coefficients sont définis par :

$$\forall j \in 1, ..., n_b, \frac{\partial c_i}{\partial x_j} = \begin{cases} 0 \text{ si } j \notin \{i, i-1\} \\ 2(x_i - x_{i-1}) \text{ si } j = i \\ -2(x_i - x_{i-1}) \text{ si } j = i-1 \end{cases}$$

On en déduit alors la forme de la jacobienne a_x de c pour $(x_1, ...x_{n_n})$:

$$a_x = \begin{pmatrix} 2(x_1 - x_0) & 0 & \cdots & \cdots & 0 \\ -2(x_2 - x_1) & 2(x_2 - x_1) & 0 & \cdots & \vdots \\ 0 & \ddots & \ddots & 0 & 0 \\ \vdots & \vdots & & -2(x_{n_b-1} - x_{n_b-2}) & 2(x_{n_b-1} - x_{n_b-2}) \\ 0 & 0 & \cdots & 0 & -2(x_{n_b} - x_{n_b-1}) \end{pmatrix}$$

On peut raisonner de manière analogue pour $(y_1, ...y_{n_n})$ et on obtient :

$$a_y = \begin{pmatrix} 2(y_1 - y_0) & 0 & \cdots & \cdots & 0 \\ -2(y_2 - y_1) & 2(y_2 - y_1) & 0 & \cdots & \vdots \\ 0 & \ddots & \ddots & 0 & 0 \\ \vdots & \vdots & & -2(y_{n_b-1} - y_{n_b-2}) & 2(y_{n_b-1} - y_{n_b-2}) \\ 0 & 0 & \cdots & 0 & -2(y_{n_b} - y_{n_b-1}) \end{pmatrix}$$

Enfin, la jacobienne a de c pour $xy = (x_1, ...x_{n_n}, y_1, ...y_{n_n})$ s'obtient en rassemblant les deux matrices par bloc :

$$a = (a_x | a_y)$$

2.5.5 Calcul de `hl`

`hl` correspondant au hessien du lagrangien en `xy` et `lm`, on peut écrire :

$$hl = \nabla^2 l(xy, \lambda) = \nabla^2 e(xy) + \sum_{i=1}^{n_b} \lambda_i \nabla^2 c_i(xy)$$

Il s'agit donc d'une matrice carrée de taille $2n_n$ par $2n_n$.

Or, on sait que e est une forme linéaire d'ordre 1. On en déduit que son hessien est forcement nul :

$$\nabla^2 e(xy) = 0$$

On est donc ramené au calcul de $\nabla^2 c_i(xy)$ pour tout $i \in \{1, ..., n_b\}$. Par la forme de la fonction c_i, on sait déjà que les dérivées croisées avec un terme d'abscisse et un terme d'ordonnée sont nuls. Il suffira donc de se concentrer sur le calcul des dérivées croisées selon les abscisses et les ordonnées séparement. De plus, les deux cas étant analogues, on se ramène au calcul du hessien de c_i en $(x_1, ...x_{n_n})$, noté $\nabla^2 c_i^x(xy)$.

Or, on sait qu'il s'agit de la matrice dont les coefficients $(k, l) \in \{1, .., n_n\}^2$ sont définis par :

$$(\nabla^2 c_i^x(xy))_{k,l} = \frac{\partial^2 c_i}{\partial x_k \partial x_l} = \begin{cases} 2 \text{ si } (k,l) = (i,i) \\ 2 \text{ si } (k,l) = (i-1, i-1) \\ -2 \text{ si } (k,l) = (i, i-1) \\ -2 \text{ si } (k,l) = (i-1, i) \end{cases}$$

Ainsi, on peut en déduire la forme de $\nabla^2 c_i^x(xy)$ dans les cas ci-dessous :

$$\nabla^2 c_1^x(xy) = \begin{pmatrix} 2 & \\ & 0 \end{pmatrix} \text{ pour } i = 1$$

$$\nabla^2 c_i^x(xy) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & -2 & 0 \\ 0 & -2 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ pour} i \in \{2, n_n - 1\}$$

$$\nabla^2 c^x_{n_n}(xy) = \begin{pmatrix} 0 & \\ & 2 \end{pmatrix} \text{ pour } i = n_n$$

On construit de la même manière $\nabla^2 c^y_i(xy)$. Enfin, $\nabla^2 c_i(xy)$ est obtenue par construction par blocs :

$$\nabla^2 c_i(xy) = \begin{pmatrix} \nabla^2 c^x_i(xy) & 0 \\ 0 & \nabla^2 c^y_i(xy) \end{pmatrix}$$

On obtient enfin `hl` par sommation :

$$hl = \sum_{i=1}^{n_b} \lambda_i \nabla^2 c_i(xy)$$

2.6 L'initialisation `ch.m` et tests

Afin de s'assurer de la correction de notre simulateur, il nous faut encore programmer l'initialisation et réaliser des tests.

2.6.1 L'initialisation `ch.m`

L'initialisation est dans notre cas qu'un script très simple nous servant à déclarer nos variables d'expérience. Nous nous en servirons dans le suite pour réaliser nous tests sur le cas test 1 proposé dans l'énoncé :

```
    L = [0.7 0.5 0.3 0.2 0.5]'
(A,B) = (1,-1)
   xy = [ 0.2 0.4 0.6 0.8 -1.0 -1.5 -1.5 -1.3 ]'
```

On représente alors la chaîne :

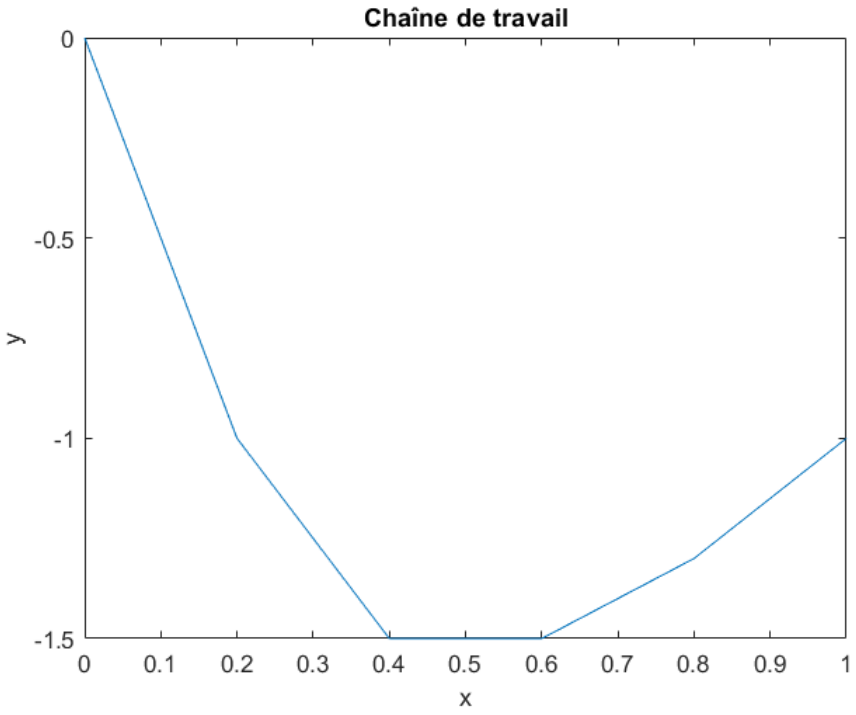


FIGURE 2 – Représentation de la chaîne de travail pour les tests

2.6.2 Test des calculs de `e` et de `c`

On peut considérer que la chaîne de travail est assez simple pour pouvoir réaliser manuellement les calculs de `e` et `c` puis les vérifier dans par le simulateur. On obtient alors les résultats :

```
e_manuel = -2.2800 et c_manuel = [0.5500,0.0400,-0.0500,0.0400,-0.1200]'
```

égaux aux résultats obtenus par le simulateur. On peut donc en conclure que celui-ci calcule bien ces valeurs.

2.6.3 Tests sur l'exactitude des dérivées

Nous allons enfin vérifier l'exactitude du calcul de notre vecteur `g` qui est le gradient de `e` en `xy`. Pour ce faire, on utilise le quotient différentiel suivant :

$$\forall i \in \{1, ..., 2n_n\}, \frac{e(x + t_i v^i) - e(x)}{t_i} = \frac{\partial e}{\partial x_i}(x) + O(t_i)$$

où v^i désigne le i^{eme} vecteur de la base canonique de $\mathbb{R}^{2n_n}$ et $t_i = \varepsilon^{1/2} \max(\tau_i, |x_i|)$ avec τ_i un vecteur de l'ordre des x_i, pris ici égal à 1 et ε l'epsilon machine.

On trouve alors dans notre cas les vecteurs de pas `pas`, le vecteur gradient `g_manuel` obtenu par le quotient différentiel à comparer au vecteur `g` et le vecteur des erreures absolues pour les abscisses et relatives pour les ordonnées `erreur` entre les vecteurs `g_manuel` et `g`.

```
     pas = 1.0e-07 * [0.1490 0.1490 0.1490 0.1490 0.2235 0.2235 0.1937]'
g_manuel = [0 0 0 0 0.6000 0.4000 0.2500 0.3500]'
       g = [0 0 0 0 0.6000 0.4000 0.2500 0.3500]'
  erreur = 1.0e-07 * [0 (abs) 0 (abs) 0 (abs) 0 (abs) ...
        0.0993 (rel) 0.0993 (rel) 0.7947 (rel) 0.3668 (rel)]'
```

On remarque donc que les erreurs relatives sont toutes au plus de l'ordre de 10^{-8} ce qui est excellent. Cela signifie le calcul des valeurs du gradient `g` par notre simulateur sont correctes.

2.6.4 Autres sorties

On affiche finalement les sorties `a` et `hl` pour le cas test 1 :

```
a =

   (1,1)       0.4000
   (2,1)      -0.4000
   (2,2)       0.4000
   (3,2)      -0.4000
   (3,3)       0.4000
   (4,3)      -0.4000
   (4,4)       0.4000
   (5,4)      -0.4000
   (1,5)      -2.0000
   (2,5)       1.0000
   (2,6)      -1.0000
   (4,7)      -0.4000
   (4,8)       0.4000
   (5,8)      -0.6000
```

FIGURE 3 – Valeur de `a` pour le cas test 1

```
hl =

   (1,1)       1.8600
   (2,1)      -0.8446
   (1,2)      -0.8446
   (2,2)       1.8826
   (3,2)      -1.0380
   (2,3)      -1.0380
   (3,3)       2.2692
   (4,3)      -1.2312
   (3,4)      -1.2312
   (4,4)       2.9860
   (5,5)       1.8600
   (6,5)      -0.8446
   (5,6)      -0.8446
   (6,6)       1.8826
   (7,6)      -1.0380
   (6,7)      -1.0380
   (7,7)       2.2692
   (8,7)      -1.2312
   (7,8)      -1.2312
   (8,8)       2.9860
```

FIGURE 4 – Valeur de `hl` pour le cas test 1

3 Optimiseur et méthode de Newton (TP2)

Dans un second temps, nous allons nous intéresser à la programmation générique d'un optimiseur s'appuyant sur la méthode de Newton. Celui-ci, à défaut de trouver un minimum pour notre problème, nous permettra néanmoins de trouver un point stationnaire où s'annule le gradient de l'énergie potentielle ce qui est une condition nécessaire pour être un point de minimum.

La détermination d'un point stationnaire de e consiste en la détermination d'une solution primale-duale $(xy_*, \lambda_*) \in \mathbb{R}^n \times \mathbb{R}^m$ de son système d'optimalité du premier ordre :

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

ou encore :

$$F(z) = 0 \text{ avec } F : \begin{cases} \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m \\ z_* = (x, \lambda) \mapsto \begin{pmatrix} \nabla e(xy) + c'(xy)^{\mathbb{T}} \lambda \\ c(xy) \end{pmatrix} \end{cases}$$

On peut alors monter que l'algorithme de Newton sous forme simplifiée consiste à calculer les valeurs successives d'une suite $z_k := (xy_k, \lambda_k)$ de telle façon à ce qu'a partir de z_k, z_{k+1} vérifie :

$$xy_{k+1} = xy_k + d_k \text{ et } \lambda_{k+1} = \lambda_k^{PQ}$$

avec,

$$\begin{pmatrix} L_k & A_k^{\mathbb{T}} \\ A_k & 0 \end{pmatrix} \begin{pmatrix} d_k \\ \lambda_k^{PQ} \end{pmatrix} = - \begin{pmatrix} \nabla e(xy_k) \\ c(xy_k) \end{pmatrix}$$

où

$$L_k = \nabla^2 l_{xx}(xy_k, \lambda_k) \text{ et } A_k = c'(xy_k)$$

L'initialisation de z_1 pourra se faire par le choix arbitraire d'un $xy_1 \in \mathbb{R}^n$ et d'un $\lambda_1 \in \mathbb{R}^m$ ou comme on le verra dans la suite d'un λ_1 permettant au solveur de ne pas itérer dans le cas où le xy_1 choisit sera déjà stationnaire.

Les itérés z_k successifs seront calculés jusqu'à la validation d'un test d'arrêt double : soit k aura dépassé une valeur maximale d'itération, soit les itérés z_k seront considérés comme suffisamment proches de la solution ce qui se caractérise par le fait que $||\nabla e(xy_k) + c'(x_k)^{\mathbb{T}} \lambda_k||_\infty$ et $||c(xy_k)||_\infty$ soient inférieures à deux bornes leur étant respectives. On renverra alors la valeur du dernier itéré z_k calculé.

3.1 Vitesse de convergence

Dans le cas de la convergence de la suite des itérés (z_k) vers une solution z_* de notre système d'optimalité, il serait intéressant de pouvoir évaluer la vitesse de convergence de notre algorithme. Celle-ci peut se caractériser par le quotient :

$$\frac{||z_{k+1} - z_*||_\infty}{||z_k - z_*||_\infty^\alpha}$$

où $\alpha \in \mathbb{N}$ dans la mesure où l'erreur $||z_k - z_*||_\infty$ ne s'annule pas ce qui est le cas ici au vu de nos tests d'arrêts.

Cependant, la méconnaissance de la solution z_* nous oblige à résoudre deux fois le problème : une première fois pour en déterminer une approximation et une seconde fois pour réaliser l'examen des quotients ce qui s'avère coûteux en calcul et donc en temps. Une autre possibilité serait la mémorisation de tous les itérés calculés en considérant le dernier itéré comme la solution mais cette fois cette méthode s'avère très coûteuse en mémoire.

Or, en considérant la convergence de (z_k) vers z_*, on peut ramener notre étude à une échelle locale dans un voisinage V de z_* dans lequel les itérés seront à partir d'un k assez grand. De par sa forme, F est différentiable en z_* et même de classe C^1 sur le voisinage V de z_* et on peut écrire le développement limité de F en z_* :

$$F(z) = F(z_*) + F'(z_*) \cdot (z - z_*) + o(||z - z_*||_\infty)$$

Or, par définition, z_* étant solution du système, $F(z_*) = 0$ et donc :

$$F(z) = F'(z_*) \cdot (z - z_*) + o(||z - z_*||_\infty)$$

On en déduit alors que sur le voisinage V :

$$||F(z)||_\infty = O(||z_- z_*||_\infty) \quad (1)$$

De plus, par inversibilité de $F'(z_*)$, on a également sur V :

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

dont on peut déduire que :

$$||z_- z_*||_\infty = O(||F(z)||_\infty) \quad (2)$$

Ainsi, on peut en conclure par (1) et (2) qu'il existe une constante $C > 0$ telle que sur le voisinage V de z_* :

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

Or, les itérés z_k étant par convergence dans le voisinage V pour k assez grand, on en déduit qu'il existe un indice k_0 à partir :

$$\forall k \geq k_0, C^{-1}||F(z_k)||_\infty \leq ||z_k - z_*|| \leq C||F(z_k)||_\infty$$

On en déduit alors que l'étude de la vitesse de convergence de l'algorithme peut se ramener à l'étude de la vitesse de convergence de la suite $(||F(z_k)||_\infty)$.

Ce fait va nous permettre d'étudier la vitesse de convergence par le biais de cette nouvelle suite dont la limite est nulle par continuité :

$$\lim_{k \to +\infty} (||F(z_k)||_\infty) = ||F(z_*)||_\infty = 0$$

ce qui nous permet de résoudre les problèmes de coûts en temps et en mémoire des méthodes précédentes. On pourra par ailleurs remarquer que ces études peuvent être réalisées avec n'importe quelles normes de $\mathbb{R}^n \times \mathbb{R}^m$ par équivalence des normes en dimension finie.

3.2 Choix du multiplicateur de lagrange initial

Il peut arriver que l'algorithme doive tourner sans précision de la valeur initiale λ_0. Celle-ci pourra alors être choisie de manière à augmenter l'efficacité de l'algorithme. Si x_0 est déjà fixé, on peut en effet choisir une valeur de λ_0 permettant au solveur de ne pas itérer dans le cas où le x_0 choisit est déjà stationnaire. Cela revient à choisir un λ_0 tel que la condition d'arrêt de convergence est vérifiée, ce qui équivaut à choisir λ_0 comme multiplicateur optimal vérifiant ainsi :

$$\nabla e(x_0) + c'(x_0)^{\mathbb{T}} \lambda_0 = 0$$

Cependant, le nombre d'inconnues étant supérieur au nombre d'équations, la matrice $c'(x_0)^{\mathbb{T}}$ n'est pas inversible. En effet, $c'(x_0)^{\mathbb{T}}$ est de taille $n \times m$ et on a en général $n > m$. Ainsi, on n'a pas garantie de trouver une solution unique à ce système. Sans la surjectivité de $c'(x_0)^{\mathbb{T}}$, on peut tout de même trouver un multiplicateur acceptable en résolvant le problème quadratique suivant :

$$\min_{\lambda_1 \in \mathbb{R}^m} (||\nabla e(x_0) + c'(x_0)^\top \lambda_0||_2^2)$$

Dont l'existence d'une solution est assurée par le théorème de Weierstrass. On peut en considérant que la norme utilisée est la norme euclidienne, on calcule le problème quadratique sous forme canonique :

$$\begin{aligned}
||c'(x_0)^\top \lambda + \nabla e(x_0)||^2 &= (c'(x_0)^\top \lambda + \nabla e(x_0))^\top \times (c'(x_0)^\top \lambda + \nabla e(x_0)) \\
&= \lambda^\top c'(x_0) c'(x_0)^\top \lambda + \lambda^\top c'(x_0) \nabla e(x_0) + \nabla e(x_0)^\top c'(x_0)^\top \lambda + \nabla e(x_0)^\top \nabla e(x_0) \\
&= \frac{1}{2} \lambda^\top (2c'(x_0) c'(x_0)^\top) \lambda + (2c'(x_0) \nabla e(x_0))^\top \lambda + \nabla e(x_0)^\top \nabla e(x_0)
\end{aligned}$$

On est donc rammené à la résolution du problème :

$$\min_{\lambda} \frac{1}{2}\lambda^\top H\lambda + f^\top \lambda$$

avec :

— $H = 2c'(x_0)c'(x_0)^\top$

— $f = 2c'(x_0)\nabla e(x_0)$

Ceci est possible en Matlab via la fonction `Quadprog` :

```
lambda_1 = quadprog(2*a*a',2*a*g)
```

3.3 Applications numériques

On applique par suite notre algorithme pour les cas tests $2a, 2b, 2c$ et $2d$ tels que les longueurs respectives des barres y soient 0.7, 0.5, 0.3, 0.2 et 0.5, que les points de fixation de la chaîne aux extrémités y soient l'origine du repère $(0, 0)$ et $(a, b) = (-1, 1)$ et tels que les positions initiales des noeuds intermédiaires soient :

Cas test	Noeuds intermédiaires
$2a$	$(0.2, -1), (0.4, -1.5), (0.6, -1.5), (0.8, -1.3)$
$2b$	$(0.2, 1), (0.4, 1.5), (0.6, 1.5), (0.8, 1.3)$
$2c$	$(0.2, -1), (0.4, -1.5), (0.6, 1.5), (0.8, -1.3)$
$2d$	$(0.2, -1), (0.4, -1.5), (0.6, 1.5), (0.8, -1.3)$

On représente ci-dessous leur aspect :

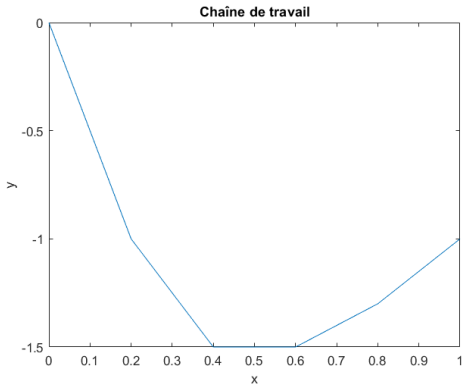


FIGURE 5 – Test 2a

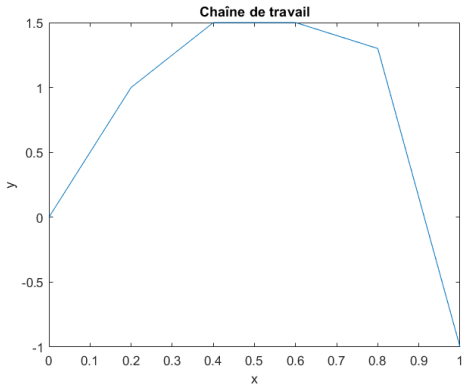


FIGURE 6 – Test 2b

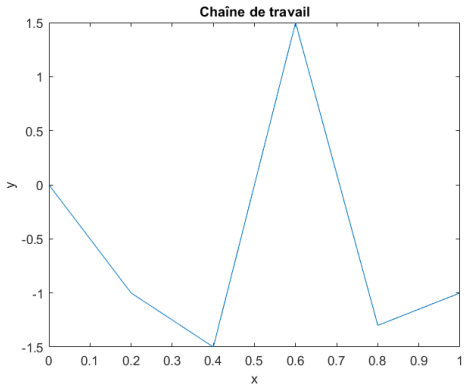


FIGURE 7 – Test 2c

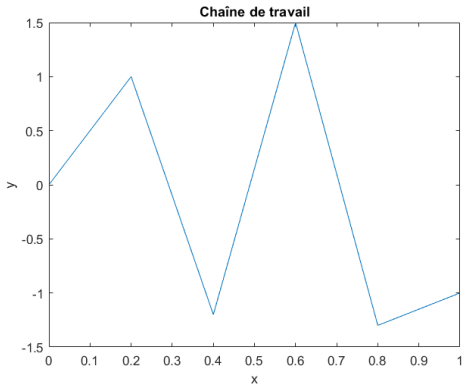


FIGURE 8 – Test 2d

FIGURE 9 – Chaînes de travail initiales $2a, 2b, 2c$ et $2d$

Après application dans chacun des cas de l'optimiseur initialisé avec son xy_1 respectif et le λ_1 correspondant, en prenant au maximum 1000 itérations et des bornes pour $||F(z_*)||_\infty$ et $||c(xy_*)||_\infty$ valant 10^{-6}, on comptabilise le nombre d'itérations effectuées ainsi que les valeurs de $||F(z_*)||_\infty$ et de $||c(xy_*)||_\infty$ qui caractérisent la convergence. On résume l'ensemble de ces données dans ce tableau :

Test	Nombre d'itération	$\|\|F(z_*)\|\|_\infty$	$\|\|c(xy_*)\|\|_\infty$
$2a$	5	$5 \cdot 10^{-7}$	$7 \cdot 10^{-8}$
$2b$	8	$3 \cdot 10^{-10}$	$6 \cdot 10^{-11}$
$2c$	12	$3 \cdot 10^{-7}$	$1 \cdot 10^{-8}$
$2d$	242	$4 \cdot 10^{-9}$	$1 \cdot 10^{-10}$

L'aspect des chaînes en sortie est quant-à lui affiché ci-dessous :

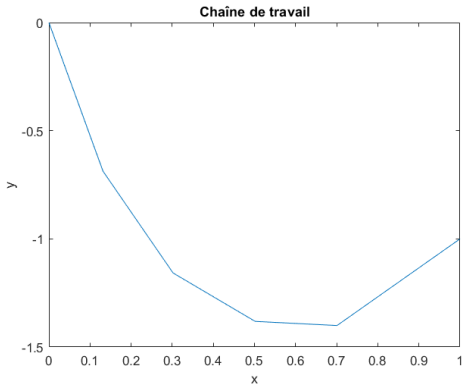


FIGURE 10 – Test 2a

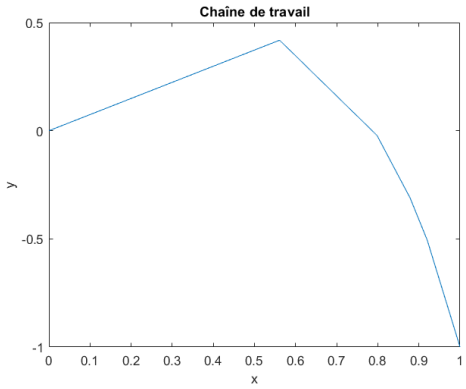


FIGURE 11 – Test 2b

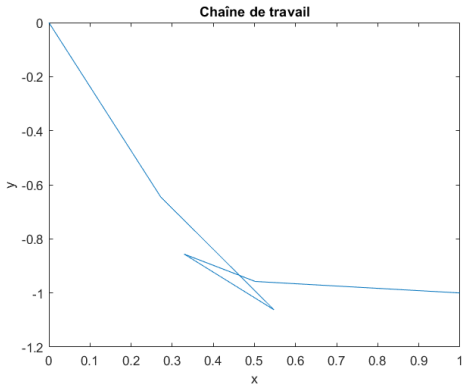


FIGURE 12 – Test 2c

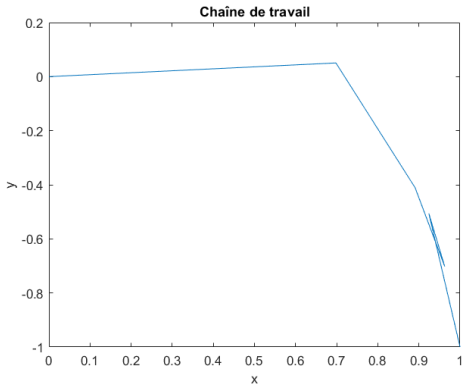


FIGURE 13 – Test 2d

FIGURE 14 – Chaînes de travail optimisées $2a, 2b, 2c$ et $2d$

On remarque que, dans nos quatre exemples, l'algorithme converge vers une solution en moins de 300 itérations et que les valeurs de $||F(z_*)||_\infty$ et $||c(xy_*)||_\infty$ sont bien toutes d'un ordre en dessous de 10^{-6} ce qui est valide avec nos conditions d'arrêt de convergence.

Cependant, on remarque bien que chaque solution approximée semble différente. En effet, même si le problème initial de minimisation de l'énergie potentielle e admet comme montré précédemment une solution unique, il peut y avoir plusieurs points stationnaires au problème. Cependant, il est possible que la solution se trouve parmi les configurations stables affichées ci-dessus.

On peut stipuler que la solution est un minimum ce qui nous permettera d'éliminer parmi les points stationnaires trouvés ce qui ne peuvent être solution de notre problème de minimisation. Or, en calculant les hessiennes $hl = \nabla^2 l(xy_*, \lambda_*)$ et en déterminant leurs valeurs propres grâce à la fonction `eig` de Matlab, on remarque que :

- Pour $2a$: Les valeurs propres sont positives donc il s'agit d'un minimum ($\min(Sp(hl)) = 0.4681$ et $\max(Sp(hl)) = 4.8135$ où $Sp(hl)$ désigne le spectre de hl).
- Pour $2b$: Les valeurs propres sont négatives donc il s'agit d'un maximum ($\min(Sp(hl)) = -14.3120$ et $\max(Sp(hl)) = -0.5748$).
- Pour $2c$ et $2d$: Les valeurs propres sont positives et négatives donc il s'agit de deux points selles ($\min(Sp(hl)) = -4.3087$ et $\max(Sp(hl)) = 8.2467$ pour 2c et $\min(Sp(hl)) = -6.8871$ et $\max(Sp(hl)) = 9.6828$ pour 2d).

Ces observations, notamment dans le cas de $2a$ et $2b$, semblent cohérentes avec l'aspect des chaînes optimisées. La chaîne optimisée du cas test $2a$ est donc la seule pouvant être solution du problème de minimisation. Nous ne pouvons néanmoins pas conclure sur son identité réelle.

Enfin, il serait intéressant de se pencher sur l'efficacité de l'algorithme de Newton dans ce cas d'utilisation, celui-ci étant théoriquement de vitesse de convergence quadratique. Or, par ce qui précède, la vitesse de convergence peut être étudiée par l'étude du rapport :

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

On calcule ce rapport pour chaque itération de l'algorithme dans le cas test $2a$ et on obtient alors le vecteur `Quot_list` tel que :

```
Quot_list =

    1.8182
    0.9924
    0.2407
    0.0583
    0.0011
```

FIGURE 15 – Evolution du rapport $||F(z_{k+1})||_\infty / ||F(z_k)||_\infty^2$ de la cas test $2a$

On remarque alors que le rapport semble converger vers 0. La théorie retenti dans la pratique et on voit bien que la vitesse de convergence de l'algorithme de Newton est ici quadratique.

4 Globalisation par recherche linéaire (TP3)

L'algorithme de Newton souffre néamoins d'une limite : si le point d'initialisation est trop éloigné d'un extremum local, l'algorithme diverge. On aimerait donc l'améliorer en forçant sa convergence même dans le cas où l'itéré initial serait plus éloigné d'un point stationnaire, indépendamment de la recherche d'un minimum global.

4.1 Nouvelle méthode de descente

Pour ce faire, on peut user d'une technique de globalisation par recherche linéaire. Celle-ci consiste à se rammener à une fonction réelle dite de mérite, atteignant un minimum pour une solution de notre problème de recherche de zéro de la fonction F. On pourra par exemple prendre la fonction des moindres carrés φ définie par :

$$\varphi : \begin{cases} \mathbb{R}^{n+m} \to \mathbb{R} \\ z \mapsto \frac{1}{2}||F(z)||_2^2 \end{cases}$$

On pourra alors forcer la convergence de l'algorithme de Newton en modifiant la méthode de calcul des directions de descente $p_{k+1} = (x_{k+1}, \lambda_{k+1})$ de F en z_k à partir de $p_k = (x_k, \lambda_k)$.

En effet, on calculera desormais à chaque itération un nouvel itéré $\alpha_k > 0$ de sorte que :

$$xy_{k+1} = xy_k + \alpha_k d_k \text{ et } \lambda_{k+1} = \lambda_k + \alpha_k \mu_k$$

avec d_k et μ_k définis par :

$$\begin{pmatrix} L_k & A_k^{\mathbb{T}} \\ A_k & 0 \end{pmatrix} \begin{pmatrix} d_k \\ \mu_k \end{pmatrix} = - \begin{pmatrix} \nabla e(xy_k) + A_k^{\mathbb{T}} \lambda_k \\ c(xy_k) \end{pmatrix}$$

Dans ce cas, la direction de descente de l'algorithme de Newton $p_k := -F'(z_k)^{-1} F(z_k)$ est également une direction de descente en z_k de la fonction φ car, par définition, F étant infiniment dérivable, φ est en particulier différentiable et on a :

$$\varphi'(z_k) \cdot p_k = F(z_k)^{\mathbb{T}} F'(z_k) \cdot p_k = -||F(z_k)||_2^2 < 0 \text{ pour } F(z_k) \neq 0$$

4.2 Choix des α_k et règle d'Armijo

La valeur des α_k ne peut quand à elle être totallement arbitraire. Premièrement, on peut facilement comprendre que pour que $\alpha_k p_k$ reste une direction de descente stricte de φ, il faut que $\underline{\alpha_k > 0}$ pour α_k suffisamment petit. La valeur de α_k ne peut être trop grande car quand bien même p_k est une direction de descente, un déplacement trop grand de z_k à z_{k+1} peut engendrer une sortie de l'itéré de son voisinage de convergence vers la solution la plus proche ce qui peut entrainer la génération d'une suite d'itérés ératique. Ayant cependant certitude que p_k est bien une direction de descente de φ en z_k, on sait que $\varphi(z_k + \alpha_k p_k) < \varphi(z_k)$ pour $\alpha_k \in]0, 1]$. De plus, $\alpha_k > 0$ ne peut être choisi trop petit car pour $\varepsilon > 0$ tel que :

$$\forall k \geq 1, \begin{cases} 0 < \alpha_k \leq \frac{\varepsilon}{2^k ||p_k||} \\ e(z_k + \alpha_k p_k) < e(z_k) \end{cases}$$

On trouve par récurrence que pour tous $i, j \geq 1$

$$z_i - z_j = \sum_{k=j}^{i-1} \alpha_k p_k$$

Donc,

$$||z_i - z_j|| \leq \sum_{k=j}^{i-1} \alpha_k ||p_k|| \leq \varepsilon \sum_{i \geq l} \frac{1}{2^k} = \frac{\varepsilon}{2^{j-1}} \underset{j \to +\infty}{\to} 0$$

Ainsi, si α_k est trop petit, z_k est une suite de Cauchy dans un espace de Banach donc z_k converge et ce vers $\bar{z} \in \mathscr{B}(z_1, \varepsilon)$ (en prenant $j = 0$ dans l'inégalité ci-dessus). Ainsi, la valeur de $\varepsilon < 0$ étant aussi petite que souhaité, il existe des configurations où si z_1 n'est pas solution, aucune solution ne sera dans $\mathscr{B}(z_1, \varepsilon)$ et l'algorithme ne pourra alors pas converger vers une solution.

Afin de résoudre ces problèmes, Armijo à travaillé sur une règle permettant de garantir un choix pertinent des α_k. Sans considérer l'interpolation, celle-ci consiste à prendre un $\alpha_k > 0$ vérifiant :

$$\varphi(z_k + \alpha_k p_k) \leq \varphi(z_k) + \omega \alpha_k \varphi'(z_k) \cdot p_k$$

avec un ω fixé dans $]0, \frac{1}{2}[$ que l'on prendra égal à 10^{-4} (le fait de prendre ω dans un tel interval permettant la convergence de l'algorithme pour $\alpha_k = 1$ lorsque les x_k sont suffisamment proche d'une solution). Pour ce faire, on choisit un α_k^1 initial égal à 1, ce qui permet de garantir que l'on ne soit ni dans le cas d'une valeur trop faible, ni dans le cas d'une valeur trop forte. On divise ensuite successivement α_k^1 par 2 de sorte à créer à chaque itération k de l'algorithme de Newton une suite d'itérés (α_k^i) telle que :

$$\forall i, \alpha_k^i = \frac{1}{2^{i-1}}$$

On choisira alors α_k comme le premier terme α_k^i vérifant la règle d'Armijo.

Enfin, afin d'éviter toute boucle infinie, nous devons nous assurer de l'existence d'un $\alpha_k^i > 0$ vérifiant la règle d'Armijo. Si on suppose par l'absurde que ce n'est pas le cas, on sait alors qu'il existe une suite (α_k^i) de termes strictement positifs convergent vers 0 pour $i \to 0$ tels que :

$$\forall i \geq 0, \varphi(z_k + \alpha_k^i p_k) > \varphi(z_k) + \omega \alpha_k^i \varphi'(z_k) \cdot p_k$$

Soit,

$$\forall i \geq 0, \frac{\varphi(z_k + \alpha_k^i p_k) - \varphi(z_k)}{\alpha_k^i} > \omega \varphi'(z_k) \cdot p_k$$

Et en passant à la limite pour $i \to 0$:

$$\varphi'(z_k) \cdot p_k > \omega \varphi'(z_k) \cdot p_k$$

Ce qui implique que :

$$\varphi'(z_k) \cdot p_k > 0$$

ce qui est absurde, p_k étant une direction de descente de φ en z_k. Ainsi, on en déduit bien que pour tout $k \geq 1$, il existe bien un α_k^i vérifiant la règle d'Armijo et l'algorithme de recherche est donc de finitude garantie.

4.3 Applications numériques

On décide alors d'appliquer notre nouvelle algorithme de Newton globalisé au cas test 2d déjà rencontrés précédemment ainsi qu'aux nouveaux cas test 3a, 3b et 3c dont les données sont représentés ci-dessous :

Cas test	Noeuds	Longueur des barres
$2d$	$(0,0),(0.2,-1),(0.4,-1.5),(0.6,1.5),(0.8,-1.3),(-1,1)$	$0.7, 0.5, 0.3, 0.2, 0.5$
$3a$	$(0,0),(0.5,0.4),(1,0)$	$0.6, 0.6$
$3b$	$(0,0),(0.5,0.3),(1,0)$	$2, 1$
$3c$	$(0,0),(0.3,0.3),(0,-1)$	$2, 1$

On représente également leur aspects initiaux ci-dessous :

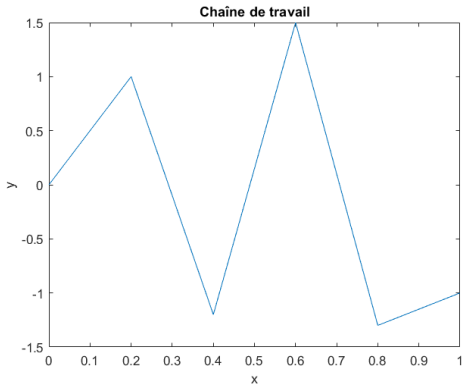


FIGURE 16 – Test 2d

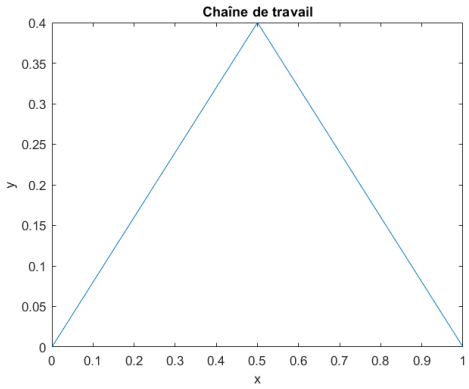


FIGURE 17 – Test 3a

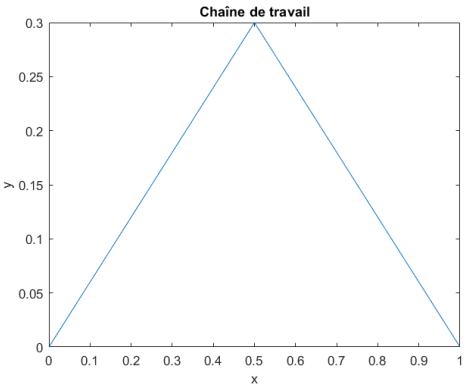


FIGURE 18 – Test 3b

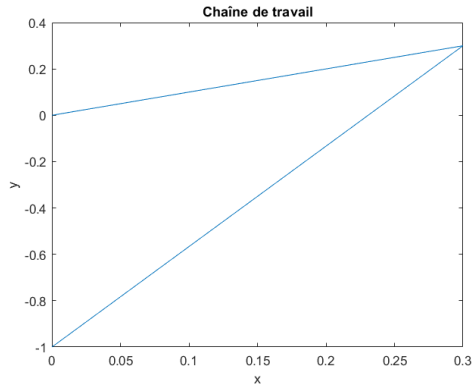


FIGURE 19 – Test 3c

FIGURE 20 – Chaînes de travail initiales $2d, 3a, 3b$ et $3c$

4.3.1 Cas test 2d

Le cas test 2d va nous permettre de comparer sur un même cas test les performances de l'algorithme de Newton avec et sans recherche linéaire sur un même cas test. Cependant, l'implémentation de la recherche linéaire et l'apparition des itérés α_k nous empèche d'utiliser la formule simplifiée (celle-ci n'étant valide que si $\alpha_k = 1, \forall k$) :

$$xy_{k+1} = xy_k + d_k \text{ et } \lambda_{k+1} = \lambda_k^{PQ}$$

avec,

$$\begin{pmatrix} L_k & A_k^{\mathbb{T}} \\ A_k & 0 \end{pmatrix} \begin{pmatrix} d_k \\ \lambda_k^{PQ} \end{pmatrix} = - \begin{pmatrix} \nabla e(xy_k) \\ c(xy_k) \end{pmatrix}$$

où

$$L_k = \nabla^2 l_{xx}(xy_k, \lambda_k) \text{ et } A_k = c'(xy_k)$$

On est donc contraint de modifier les matrices de calcul de l'itéré pour se ramener au cas plus général :

$$xy_{k+1} = xy_k + \alpha_k d_k \text{ et } \lambda_{k+1} = \lambda_k + \alpha_k \mu_k$$

avec d_k et μ_k définis par :

$$\begin{pmatrix} L_k & A_k^{\mathbb{T}} \\ A_k & 0 \end{pmatrix} \begin{pmatrix} d_k \\ \mu_k \end{pmatrix} = - \begin{pmatrix} \nabla e(xy_k) + A_k^{\mathbb{T}} \lambda_k \\ c(xy_k) \end{pmatrix}$$

Cependant, la complexification de la matrice du membre de droite entraîne plus d'erreurs de calcul avec l'ajout du terme $A_k^{\mathbb{T}}\lambda_k$ qui résultent en la divergence de l'algorithme sans recherche linéaire pour le cas test 2d. Cette divergence est néanmoins corrigée dans l'algorithme avec recherche linéaire qui permet de trouver une solution similaire à la première solution trouvée pour le cas test 2d en seulement 17 itérations :

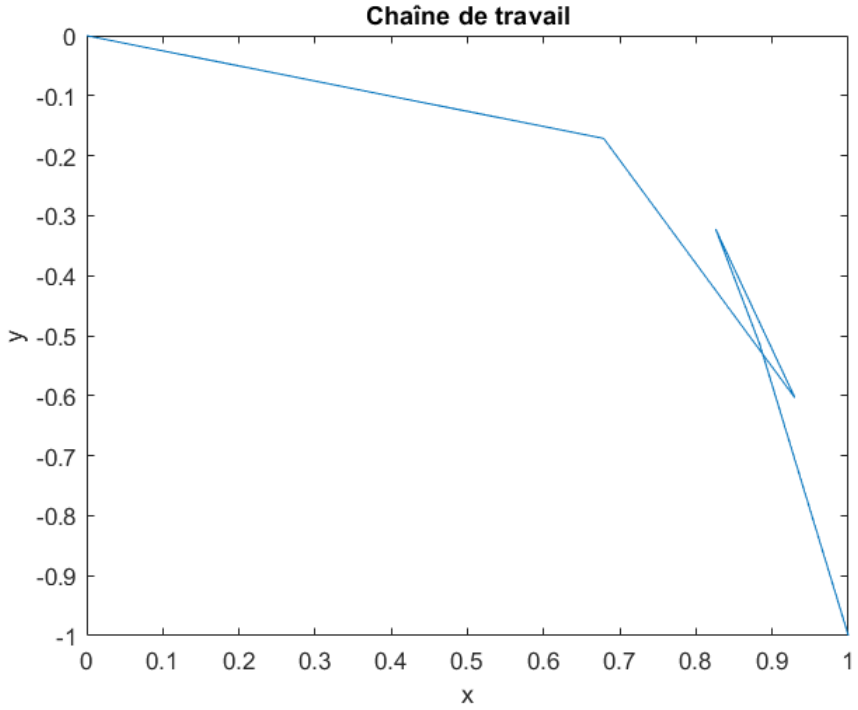


FIGURE 21 – Représentation de la chaîne solution approchée 2d par recherche linéaire

Les différences constatées avec la première solution trouvée pour le cas test 2d peuvent s'expliquer par la modification de la méthode de calcul plus générale des itérés impliquée par l'implémention de la recherche linéaire. Néanmoins, le but de notre globalisation à ici été atteint : l'algorithme converge grâce à la recherche linéaire pour un cas test où il diverge sans elle.

4.3.2 Cas test 3a

Le cas test 3a est un cas test particulier car il se trouve initialement au voisinage d'une solution. En effet, la position du noeud central correspond à une position d'équlibre instable de la chaîne tandis que les contraintes sur les longueurs sont également proches de la satisfaction. On s'attend alors à ce que l'algorithme converge très rapidement et ce sans avoir besoin de profondes recherches linéaires. Le résultat affiché ci-dessous correspond bien à nos attentes, l'algorithme de Newton convergant en 3 itérations, indépendamment de l'activation ou non de la recherche linéaire, qui de plus génère des termes α_k tous égaux à 1.

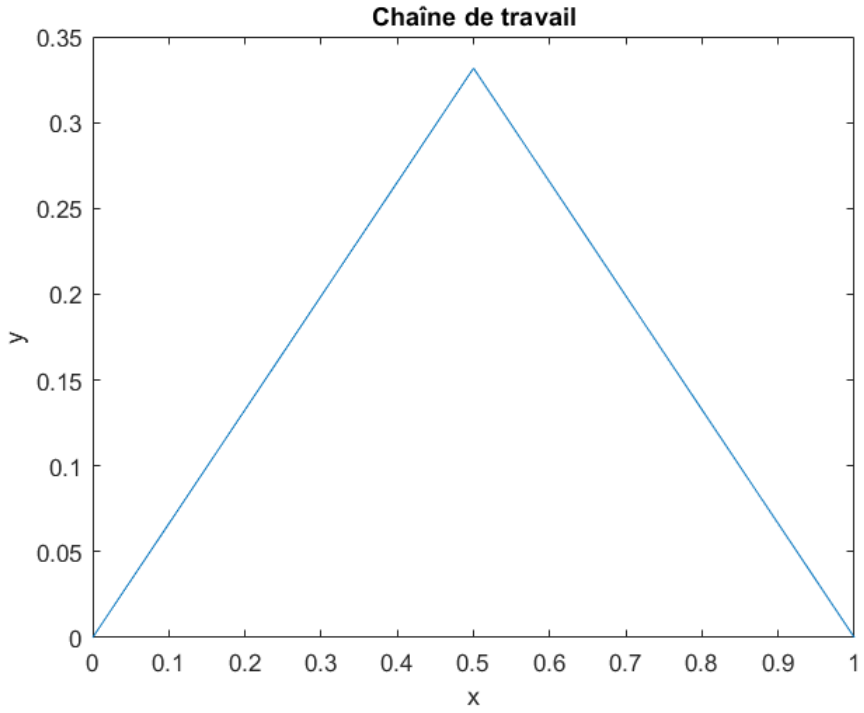


FIGURE 22 – Représentation de la chaîne solution approchée 3a par recherche linéaire

4.3.3 Cas test 3b

Le cas test 3b génère une divergence de l'algorithme avec comme sans recherche linéaire. Ceci est en réalité du à la non inversibilité de la matrice $\begin{pmatrix} L_k & A_k^{\mathbb{T}} \\ A_k & 0 \end{pmatrix}$ lors des premières itérations. On affiche ci-dessous la valeur de son déterminant pour les 20 premières itérations :

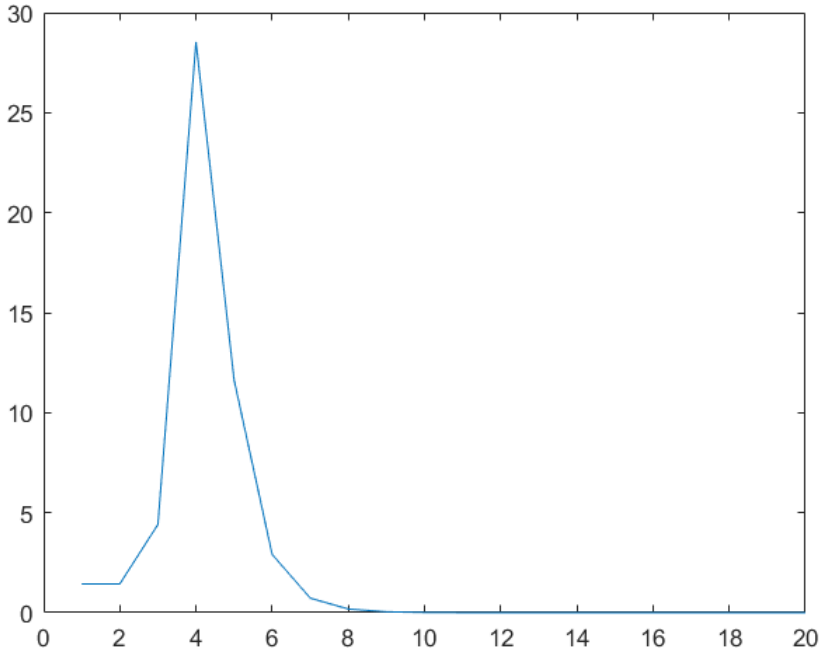


FIGURE 23 – Déterminant pour les 20 premières itérations

Les grandes valeurs du déterminant aux premières itérations amènent Matlab à générer de très importantes erreurs de calculs sur les premiers itérés ce qui s'avère désastreux quant à la convergence de l'algorithme, même si celui-ci converge rapidement vers 0 par la suite.

4.3.4 Cas test 3c

Enfin, le cas test 3c est beaucoup moins pathologique que son prédécesseur. L'algorithme converge en 14 itérations avec recherche linéaire et en 15 itérations sans recherche linaire vers la même solution affichée ci-dessous :

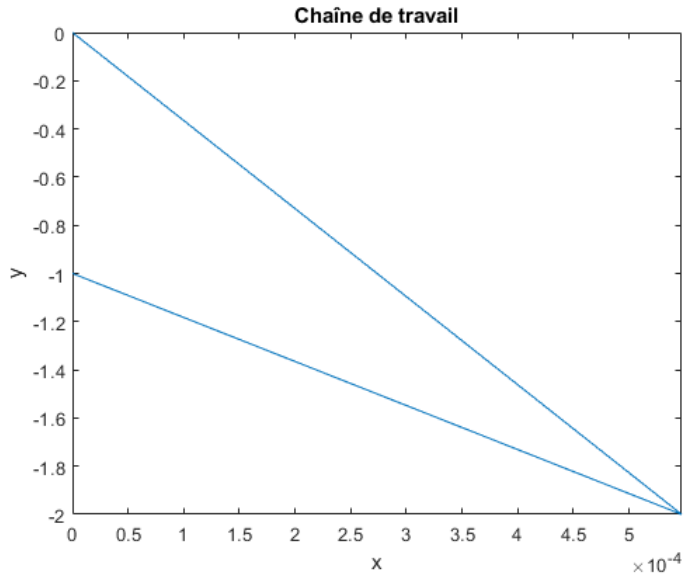


FIGURE 24 – Représentation de la chaîne solution approchée 3c par recherche linéaire

Ceci montre que la recherche linéaire peut diminuer le nombre d'itération de l'algorithme de Newton au prix d'une imbrication d'une seconde boucle interne lui correspondant.

5 Prise en compte des contraintes d'inégalité (TP4)

5.1 Nouvelle modélisation

On souhaite maintenant améliorer notre solveur pour qu'il puisse prendre en compte la présence d'un sol. Ceci correspond à l'ajout de contraintes d'inégalités que tous les noeuds de la chaîne doivent vérifier (voir figure ci-dessous) :

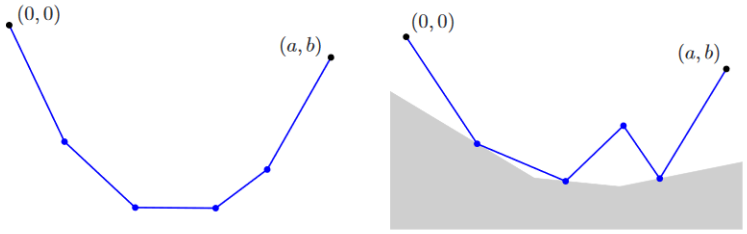


FIGURE 25 – Représentation de la chaîne avec et sans plancher

On modélise mathématiquement ce nouveau plancher par un fonction affine φ définie par :

$$\varphi : \begin{cases} \mathbb{R} \to \mathbb{R}^p \\ x \mapsto r + x * s \end{cases}$$

où r et s sont des vecteurs de $\mathbb{R}^p$ donnés.

On remarque alors que le plancher correspond dans $\mathbb{R}^2$ au complémentaire du domaine formé par l'intersection des p épigraphes des fonctions affines : $\varphi_j(x) \mapsto r_j + x * s_j$. Ainsi, dire que l'ensemble des noeuds de la chaîne sont au dessus du plancher signifie que les vecteurs r et s doivent vérifier :

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

(car les points extrêmes sont des données du problème) et que le vecteur $xy \in \mathbb{R}^n$ vérifie :

$$\forall i \in [1 : n_n], \forall j \in [1 : p], rj + x_i * s_j - y_i \leq 0$$

Cette contrainte s'ajoutant aux contraintes d'égalité sur les longueurs de barres du problème initial, quite à ajouter à la fonction c $p * n_n$ composantes, le nouveau problème se ramène à :

$$(P_{EI}) : \begin{cases} \min\limits_{xy \in \mathbb{R}^n} e(xy) \\ c_i(xy) = 0, \forall i \in [1 : n_b] \\ c_i(xy) \leq 0, \forall i \in [n_b + 1 : n_b + pn_n] \end{cases} \Leftrightarrow \min_{xy \in XY} e(xy)$$

avec :

$$\forall i \in [1 : n_n], \forall j \in [1 : p], c_{n_b+(j-1)n_n+i}(xy) = rj + x_i * s_j - y_i$$

5.2 Modification du simulateur

Après ces considérations théoriques, nous pouvons débuter l'implémentation d'une méthode numérique de résolution de notre problème P_{EI}. Nous tâcherons dans cette partie d'expliquer le nouveau script d'initialisation.

Le nouveau simulateur est une fonction `chs` qui prend en argument :

- `indic` : le pilotage de l'action du simulateur.
- `xy` : le vecteur-colonne contenant d'abord les abscisses $\{x_i\}_{i=1}^{n_n}$ des nœuds, puis leurs ordonnées $\{y_i\}_{i=1}^{n_n}$.
- `lme` : le vecteur-colonne de dimension n_b contenant les multiplicateurs de Lagrange pour les contraintes d'égalité, $\lambda_E = \{\lambda_i\}_{i=1}^{n_b}$.
- `lmi` : le vecteur-colone de dimension pn_n contenant les multiplicateurs de Lagrange pour les contraintes d'inégalité, $\lambda_I = \{\lambda_i\}_{i=n_b+1}^{n_b+pn_n}$

et renvoit en sortie :

- `e` : la valeur de l'énergie potentielle en `xy` (c'est-à-dire pour les nœuds dont les coordonnées sont dans `xy`).
- `ce` : la valeur en `xy` des contraintes sur la longueur des barres, `ce` étant un vecteur-colonne de dimension n_b, que nous noterons c_E.
- `ci` : contient la valeur des pn_n contraintes d'inégalité, c'est donc un vecteur-colone de dimension pn_n, que nous noterons mathématiquement c_I.
- `g` : le gradient de e en `xy`, c'est un vecteur-colonne de dimension $2n_n$.
- `ae` : la jacobienne des contraintes d'égalité en `xy`, une matrice de n_b lignes et $2n_n$ colonnes, que nous noterons $A_E(x) = c'_E(x)$.
- `ai` : la jacobienne des contraintes d'égalité en `xy`, une matrice de pn_n lignes et $2n_n$ colonnes, que nous noterons $A_I(x) = c'_I(x)$.

— `hl` : hessien du lagrangien en `xy` et `lm`.

— `indic` : l'indicateur des résultats de la simulation (0 dans le cas normal et 1 en cas d'erreurs sur les entrées).

Les données des longueurs `L`, du plancher `R` et `S` et des coordonnées de l'extrémité droite `(A,B)` seront communiquées en tant que variables globales issues de la fonction d'initialisation. L'extrémité gauche sera en l'origine du repère.

5.2.1 Calcul de `e`, `ce`, `g`, `ae`, `hl`

Les calculs de `e`, `ce`, `g`, `ae`, `hl` correspondent aux calculs du premier simulateur pour les contraintes d'égalités et restent par conséquent inchangés.

5.2.2 Calcul de `ci`

Par définition, on sait que :

$$\forall xy \in \mathbb{R}^n, c_I(xy) = \begin{pmatrix} r_1 + x_1 s_1 - y_1 \\ r_1 + x_2 s_1 - y_2 \\ \vdots \\ r_1 + x_{n_n} s_1 - y_{n_n} \\ r_2 + x_1 s_2 - y_1 \\ \vdots \\ r_i + x_i s_j - y_i \\ \vdots \\ r_p + x_{n_n} s_p - y_{n_n} \end{pmatrix}$$

Donc on en déduit que :

$$\forall xy \in \mathbb{R}^n, c_I(xy) = \begin{pmatrix} r_1 \\ \vdots \\ r_1 \\ r_2 \\ \vdots \\ r_2 \\ \vdots \\ r_j \\ \vdots \\ r_j \\ \vdots \\ r_p \\ \vdots \\ r_p \end{pmatrix} + \begin{pmatrix} s_1 \\ \vdots \\ s_1 \\ s_2 \\ \vdots \\ s_2 \\ \vdots \\ s_j \\ \vdots \\ s_j \\ \vdots \\ s_p \\ \vdots \\ s_p \end{pmatrix} .\times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_n} \\ x_1 \\ x_2 \\ \vdots \\ x_{n_n} \\ \vdots \\ x_1 \\ x_2 \\ \vdots \\ x_{n_n} \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_n} \\ y_1 \\ y_2 \\ \vdots \\ y_{n_n} \\ \vdots \\ y_1 \\ y_2 \\ \vdots \\ y_{n_n} \end{pmatrix} \in \mathbb{R}^{p \times n_n}$$

où $.\times$ désigne l'opérateur binaire de multiplication coordonnée par coordonnée. Or, les deux premiers vecteurs sont respectivement les résultats des opérations $K_n \times r$ et $K_n \times s$ où K_n est la matrice de taille $pn_n \times pn_n$ définie par :

$$K_n = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & & & & \vdots \\ 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & & & \vdots \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \end{pmatrix} = \begin{pmatrix} u & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & u \end{pmatrix} \text{ avec } u = \underbrace{(1, ..., 1)}_{n_n \text{ fois}}^{\mathbf{T}}$$

En Matlab, K_n est obtenue par la commande : `Kn = kron(eye(p*nn),ones(nn,1))`. Les deux derniers vecteurs X_n et Y_n sont obtenus par répétition p fois des vecteurs $x = (x_1, ..., x_{n_n})^{\mathbf{T}}$ et $y = (y_1, ..., y_{n_n})^{\mathbf{T}}$. Ceci est possible grâce à aux commandes `Xn = repmat(xy(1:nn),p,1)` et `Yn = repmat(xy(nn+1:n),p,1)` respectivement. Ainsi, on obtient finalement :

$$c_I = K_n r + K_n s \; .\times X_n - Y_n$$

5.2.3 Calcul de `ai`

Pour le calcul de `ai`, on va également procéder par distinction des dérivations selon les abscisses et les ordonnées. La jacobienne de c_I pour $(x_1, ..., x_{n_n})$ est la matrice :

$$a_{Ix} = \left(\frac{\partial c_{Ii}}{\partial x_j}\right)_{i,j\in[1:pn_n]\times[1:n_n]} = \begin{pmatrix} s_1 I_{n_n} \\ \vdots \\ s_j I_{nn} \\ \vdots \\ s_p I_{n_n} \end{pmatrix} = s. \times VI_{n_n}$$

où VI_{n_n} est une matrice de taille $pn_n \times n_n$ définie par bloc comme un vecteur colone de n_n matrices identité de taille n_n notés I_{n_n}. De manière analogue pour les ordonnées, on a :

$$a_{Iy} = \left(\frac{\partial c_{Ii}}{\partial y_j}\right)_{i,j\in[1:pn_n]\times[1:n_n]} = - \begin{pmatrix} I_{n_n} \\ \vdots \\ I_{nn} \\ \vdots \\ I_{n_n} \end{pmatrix} = -VI_{n_n}$$

La matrice VI_{n_n} est obtenue par la formule Matlab : `VI = repmat(eye(nn),p,1)` tandis que la multiplication $s. \times VI_{n_n}$ coordonnée par coordonée adaptée aux matrices définies par bloc est réalisée par `aix = bsxfun(@times,VI,Sn)`. Ainsi, on obtient finallement :

$$a_I = (a_{Ix}|a_{Iy})$$

5.3 Algortihme d'optimisation quadratique successive

L'ajout de contraintes d'inégalités dans le problème P_{EI} nous oblige à modifier notre algorthme. En effet, l'algorithme de Newton implémenté précédemment ne fonctionne que pour les cas avec seulement des contraintes d'égalités. On le remplace donc par un autre algorithme nommé algorithme d'optimisation quadratique successive (OQS). Celui-ci génère également une suite primale-duale $\{(x_k, \lambda_k)\} \subseteq \mathbb{R}^n \times \mathbb{R}^m$ d'itérés à partir d'un itiéré initial $\{(x_0, \lambda_0)\}$. Le calcul de l'itéré $\{(x_{k+1}, \lambda_{k+1})\}$ à partir de $\{(x_k, \lambda_k)\}$ se fait par le calcul de la solution primale-duale $\{(d_k, \lambda_k^{PQ})\}$ du système :

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

avec M_k une approximation du hessien du lagrangien $\nabla^2_{xx} l(x_k, \lambda_k)$ ou lui-même. Ce problème, nommé problème quadratique osculateur, est en pratique très difficle à résoudre et sera résolu dans notre cas par la fonction `quadprog` de Matlab. Le nouvel itéré sera donc calculé par recherche linéaire toujours selon la règle d'Armijo :

$$x_{k+1} = x_k + \alpha_k x_k \text{ et } \lambda_{k+1} = \lambda_k + \alpha_k(\lambda_k^{PQ} - \lambda_k)$$

On utilisera alors la condition d'arrêt :

$$\max\{||\nabla_x l(x_k, \lambda_k)||_\infty, ||c_E(x_k)||_\infty, ||\min((\lambda_k)_I, -c_I(x_k)||_\infty\} < \varepsilon$$

5.3.1 Choix de la matrice $\mathbf{M}_k$

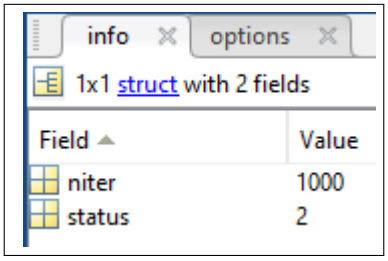
On choisira pour matrice M_k une approximation définie positive du hessien $\nabla^2_{xx} l(x_k, \lambda_k)$. Ceci est motivé 2 raisons principales :

- L'utilisation d'une matrice définie positive facilite grandement la résolution du problème quadratique osculateur : on peut passer jusqu'à une complexité de résolution polynomiale à partir d'un problème non-polynomial dans le cas où la matrice n'est pas définie positive. De plus, le problème est "mieux posé" dans la mesure où la présence d'une matrice M_k définie positive implique que le problème n'admette qu'au plus une solution.
- La solution d_k du problème sera une direction de descente d'une fonction de mérite ce qui permet la globalisation de l'algorithme par une méthode de recherche linéaire comme la méthode décrite précédemment. On pourra par exemple prendre la fonction de mérite : $\Theta_\sigma(x) = e(x) + \sigma||c(x)||_2$ avec $\sigma > 0$. On peut alors montrer que dans le cas où M_k est définie positive, x_k n'est pas un point stationnaire de P_{EI} et que $\sigma > ||\lambda_k^{PQ}||_D$ ($||\lambda||_D = \sup_{||x||_2 \leqslant 1} | < \lambda, x > |$) étant la norme duale de $||x||_2$ pour le produit scalaire euclidien), $\Theta_\sigma(x_k, \lambda_k) < 0$.

Le seul défaut de cette approximation est la perte de la convergence quadratique de l'algorithme. On peut toutefois assurer une convergence superlinéaire dans le cas où l'approximation M_k génère des directions de descentes d_k suffisamment pertinentes. La méthode d'approximation retenue est la factorisation de Cholesky modifiée, réalisée par le code `cholmod` de Matlab. Celui-ci permet de générer des matrices E_k diagonale semi-définie positive, L_k triangulaire inférieure et D_k diagonale définie positive telles que $H_k + E_k = L_k D_k L_k^\top$. L'approximation M_k sera alors : $M_k = L_k D_k L_k^\top$.

Le choix de la factorisation de Cholesky modifiée permet la génération d'approximations du hessien du lagrangien suffisamment précises pour permettre une

convergence rapide de l'algorithme. En effet, en prenant une autre matrice définie positive : l'identité I, on remarque que la convergence est considérablement ralentie et n'a toujours pas eu lieu au bout de 1000 itérations pour le cas test 4c.



Ce résultat (affiché ci-dessous) diffère par ailleurs du résultat obtenu pour 4c décrit ci-après.

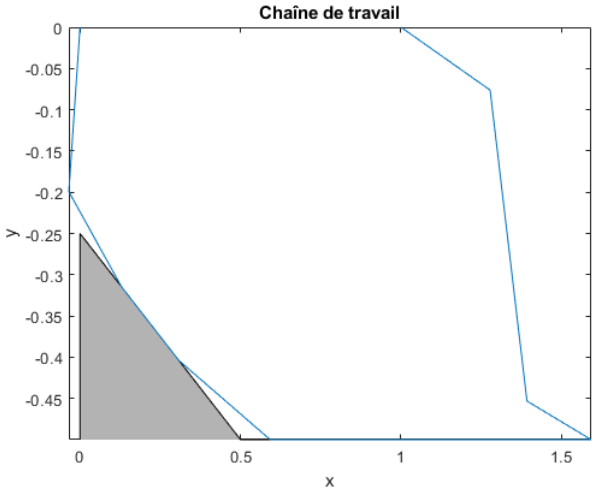


FIGURE 26 – Représentation de la chaîne (cas où $M_k = I$)

5.3.2 Choix d'un multiplicateur initial

Le choix du multiplicateur se fait avec des arguments similaires au cas avec égalité. En un point initial solution x_0 du problème suivant :

$$(P_2) \quad \begin{cases} \min\ e(\mathrm{x},\mathrm{y}) \\ c_i(\mathrm{x},\mathrm{y}) = 0, & i \in [1 : n_b] \\ c_i(\mathrm{x},\mathrm{y}) \leqslant 0, & i \in [n_b + 1 : n_b + pn_n], \end{cases}$$

On a les fonctions e, c_i (contraintes d'inégalités) et c_e (contraintes d'égalités) qui sont différentiables et par ailleurs en supposant que les contraintes du problème sont qualifiées en x_0 donc on aura l'existence d'un multiplicateur lagrangien λ_0 qui vérifiera les conditions de Karush-Kuhn-Tucker suivantes :

$$\begin{cases} \nabla f(x_0) + c'(x_0)^T \lambda_0 = 0 \\ c_E(x_0) = 0 \\ 0 \leq (\lambda_0)_I \perp c_I(x_0) \leq 0 \end{cases}$$

Or pour trouver λ_0 on peut alors résoudre un problème des moindres carrés suivants :

$$\begin{cases} min_\lambda ||c'(x_0)^T \lambda + \nabla f(x_0)||^2 \\ c_I(x_0)\lambda_I = 0 \\ \lambda_I \geq 0 \end{cases}$$

C'est un problème de moindre carrée linéaire, aussi il y a toujours existence d'une solution de ce problème, et cette solution sera ici le multiplicateur optimal trouvé par l'algorithme sans faire d'itérations. On remarque cependant la présence de contraintes supplémentaires dues aux contraintes d'inégalité de notre problème. On peut en considérant que la norme utilisée est la norme euclidienne, on calcule le problème quadratique sous forme canonique :

$$\begin{aligned} ||c'(x_0)^\top \lambda + \nabla e(x_0)||^2 &= (c'(x_0)^\top \lambda + \nabla e(x_0))^\top \times (c'(x_0)^\top \lambda + \nabla e(x_0)) \\ &= \lambda^\top c'(x_0) c'(x_0)^\top \lambda + \lambda^\top c'(x_0) \nabla e(x_0) + \nabla e(x_0)^\top c'(x_0)^\top \lambda + \nabla e(x_0)^\top \nabla e(x_0) \\ &= \frac{1}{2} \lambda^\top (2c'(x_0) c'(x_0)^\top) \lambda + (2c'(x_0) \nabla e(x_0))^\top \lambda + \nabla e(x_0)^\top \nabla e(x_0) \end{aligned}$$

On peut donc se rammener au problème quadratique équivalent :

$$\begin{cases} \min\limits_\lambda \frac{1}{2} \lambda^\top H \lambda + f^\top \lambda \\ A_{eq} \lambda = 0 \\ A\lambda \leq 0 \end{cases}$$

avec :

- $H = 2c'(x_0)c'(x_0)^\top$
- $f = 2c'(x_0)\nabla e(x_0)$
- $A_{eq} = \begin{pmatrix} 0 & 0 \\ 0 & diag(c_I(x_0)) \end{pmatrix}$
- $A = \begin{pmatrix} 0 & 0 \\ 0 & -I_{m_I} \end{pmatrix}$

Ceci s'implémente alors toujours grâce à la fonction `Quadprog` :

```
lambda_1 = quadprog(2*a*a',2*a*g,A,zeros(m,1),Aeq,zeros(m,1)
```

5.4 Critère de non-minimalité

Il peut arriver que la solution trouvée soit un point stationnaire mais pas un minimum de la fonction e. Or, on sait qu'une condition nécessaire du deuxième ordre, en cas de contraintes d'égalités, pour que le point x^* trouvé soit un minimum local est :

$$\begin{cases} x^* \text{ est un minimum local} \\ c'_e(x^*) \text{ est surjective} \end{cases} \Rightarrow \exists \lambda, \begin{cases} \nabla_x l(x^*, \lambda) = 0 \\ c_e(x^*) = 0 \\ \nabla^2_{xx} l(x^*, \lambda) \succeq 0 \end{cases}$$

On peut alors se référer à sa contraposée :

$$\forall \lambda, \begin{cases} \nabla_x l(x^*, \lambda) \neq 0 \\ \text{ou} \\ c_e(x^*) \neq 0 \\ \text{ou} \\ \neg(\nabla^2_{xx} l(x^*, \lambda) \succeq 0) \end{cases} \Rightarrow \begin{cases} x^* \text{ n'est pas un minimum local} \\ \text{ou} \\ c'_e(x^*) \text{ n'est pas surjective} \end{cases}$$

Or, en cas de convergence, en notant (x^*, λ^*) le dernier itéré donné en sortie de l'algorithme, on sait par les tests d'arrêt que :

$$\begin{cases} \nabla_x l(x^*, \lambda^*) \approx 0 \\ c_e(x^*) \approx 0 \end{cases}$$

En supposant l'unicité de λ^* pour x^*, on peut donc se ramener au critère :

$$\neg(\nabla^2_{xx} l(x^*, \lambda^*) \succeq 0) \Rightarrow \begin{cases} x^* \text{ n'est pas un minimum local} \\ \text{ou} \\ c'_e(x^*) \text{ n'est pas surjective} \end{cases}$$

Et par manipulation logique, on peut ramener le deuxième prédicat à gauche de l'implication pour obtenir le critère équivalent :

$$\begin{cases} \neg(\nabla^2_{xx} l(x^*, \lambda^*) \succeq 0) \\ \text{et} \\ c'_e(x^*) \text{ est surjective} \end{cases} \Rightarrow x^* \text{ n'est pas un minimum local}$$

On peut remplacer la propriété suffisante par la propriété équivalente plus adaptée à une implémentation sur ordinateur :

$$\begin{cases} \min(Sp(\nabla^2_{xx} l(x^*, \lambda^*))) < 0 \\ \text{et} \\ rg(c'_e(x^*)) = 2 \times n_n \end{cases} \Rightarrow x^* \text{ n'est pas un minimum local}$$

Cependant, afin d'éviter des erreurs lors de la comparaison avec 0, on préfèrera se ramener à une comparaison avec un seuil $-\varepsilon_\lambda$ de sorte que :

$$\begin{cases} \min(Sp(\nabla^2_{xx} l(x^*, \lambda^*))) < -\varepsilon \\ \text{et} \\ rg(c'_e(x^*)) = 2 \times n_n \end{cases} \Rightarrow x^* \text{ n'est pas un minimum local}$$

On pourra également ajouter une tolérance pour le calcul du rang de la matrice $c'_e(x^*)$ qu'on l'on note $\varepsilon_e > 0$. Finallement, le critère dans Matlab devient :

```
min(eig(hl))<-epsilon_lambda & rank(ae,epsilon_e) = 2*nn
```

Si ce critère est vérifié, alors x^* ne peut pas être un minimum local. Ce résultat n'est cependant plus vrai dans le cas de contraintes d'inégalités. Il peut exister en effet des minimums "locaux" au bord du domaine admissible dans le cas où des contraintes de c_I seraient actives tels que $\nabla^2_{xx} l(x^*, \lambda^*)$ n'est pas définie positive. On peut néanmoins se ramener au problème suivant :

$$(P^*_{EI}) : \begin{cases} \min\limits_{x,\lambda} e(x, y) \\ c_e(x, y) = 0 \\ c_i(x, y) = 0, i \in I^0(x_*, \lambda_*) \end{cases}$$

avec $I^0(x_*, \lambda_*)$ l'ensemble des indices des contraintes d'inégalités actives. On est donc ramené à un problème de minimisation avec contraintes d'égalité et le critère précédent reste applicable. On remarque que par la forme du hessien du lagrangien de P_{EI} :

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

Si (x_*) est un minimum, KKT nous indique l'annulation des multiplicateurs de lagrange associés aux contraintes d'inégalités non actives ce qui signifie qu'on aurait :

$$hl = \nabla^2 l(x_*, \lambda) = \nabla^2 e(x_*) + \sum_{i \in E} \lambda_i \nabla^2 c_i(x_*) + \sum_{i \in I^0} \lambda_i \nabla^2 c_i(x_*)$$

On constate alors l'égalité des hessiens des lagrangiens des deux problèmes en un minimum local x^* dans P_{EI} (l'égalité des gradients des lagrangiens sera aussi assurée). De plus, étant donné que ce $X_{EI} \subset X^*_{EI}$, on constate que si (x^*, λ^*) est effectivement un minimum local dans le cadre de P_{EI}, il l'est alors dans le cadre de P^*_{EI}. Ainsi, par contraposition, un critère de non minimalité de x^* est de vérifier qu'il n'est pas minimum pour P^*_{EI}, ce qui impliquera qu'il n'est pas minimum pour P_{EI}. On obtient alors en notant le vecteur des contraintes $c^0 = (c_e, c_{I^0})$:

$$\begin{cases} \min(Sp(\nabla^2_{xx} l(x^*, \lambda^*))) < -\varepsilon \\ \text{et} \\ rg(c^{0\prime}(x^*)) = 2 \times n_n \end{cases} \Rightarrow x^* \text{ n'est pas un minimum local}$$

Le code Matlab devient quant à lui :

```
min(eig(hl))<-epsilon_lambda & rank(a0,epsilon_e) = 2*nn
```

5.5 Convergence globale de l'algorithme OQS

Dans le cas test 4a, notre solveur utilisant l'algorithme SQP renvoie un minimum global de la chaîne tandis que le solveur sans contrainte d'inégalité ne trouvait qu'un point stationnaire (avec ou sans recherche linéaire). Aussi, on peut comprendre cela grâce à un résultat de convergence locale de OQS/SQP, prenons x_* solution de notre problème de départ :

- Tout d'abord, on a que $e(x, y) = \sum_{i=1}^{n_b - 1} y_i \frac{L_i + L_{i+1}}{2} + L_{n_b} \frac{y_{n_b}}{2}$, ainsi e est polynomiale en les coordonnées donc $e \in \mathscr{C}^{2,1}$ sur l'ensemble de définition, de plus les contraintes d'égalités et d'inégalités sont elles aussi polynomiales donc $c \in \mathscr{C}^{2,1}$.(Ceci est donc aussi valable au voisinage de la solution x_*)
- De plus, on a l'existence d'un multiplicateur lagrangien par le théorème de Karush-Kuhn-Tukker, ce dernier est unique dans le cas-test 4a)
- Enfin par le théorème de Karush-Kuhn-Tucker, on a : $\nabla_x l(x_*, \lambda_*) = 0$ et $c_E(x_*) = 0$, par ailleurs on obtient le spectre de la hessienne réduite du lagragien suivant :

```
4.10533         0         0
      0   0.61340         0
      0         0   2.03786
```

Ainsi, les valeurs propres sont ici les coefficients diagonaux, ces derniers sont tous positifs (4.10,0.61,2.04) et donc la hessienne du lagrangien est définie positive sur $N(c'(x_*))$

Ainsi, par théorème de convergence locale de OQS, il existe un voisinage V de (x_*, λ_*) tel que si $(x_1, \lambda_1) \in V$ alors l'algorithme OQS démarrant en (x_1, λ_1) :

— peut générer une suite $(x_k, \lambda_k) \subset V$ en calculant à chaque itération des points stationnaires du PQO.

— cette suite (x_k, λ_k) converge quadratiquement vers (x_*, λ_*).

Or ici x_* n'est pas un simple point stationnaire du problème mais une solution, ce qui explique que lorsque l'on part d'une position initiale dans un voisinnage de la solution, on a alors comme dans le cas-test 4a, la convergence vers ce minimum global avec l'algorithme SQP, ce qui n'était pas le cas avec l'algorithme (avec ou sans recherche linéaire) utilisé dans les cas sans contraintes.

5.6 Etude des cas tests

On applique par suite notre algorithme pour les cas tests $4a, 4b$ et $4c$ tels que le premier point de fixation de la chaîne est $(0, 0)$, de plus les informations relatives à chaque cas-tests sont décrites dans le tableau ci-après :

Cas test	Noeuds intermédiaires	Longueur des barres	(a,b)	(R,S)
4a	(0.2,1),(0.4,1.5),(0.6,1.5), (0.8,1.3)	(0.7, 0.5, 0.3, 0.2, 0.5)	(1,-1)	/
4b	(0.1,-0.5),(0.2,-0.9), (0.3,-1.2),(0.4,-1.4), (0.5,-1.5),(0.6,-1.4), (0.7,-1.2),(0.8,-0.9), (0.9,-0.5)	(0.2, 0.2, 0.2, 0.3, 0.3, 0.5, 0.2, 0.2, 0.3, 0.1)	(1,0)	(-0.25,-0.5)
4c	(0.1,-0.5),(0.2,-0.9), (0.3,-1.2),(0.4,-1.4), (0.5,-1.5),(0.6,-1.4), (0.7,-1.2),(0.8,-0.9), (0.9,-0.5)	(0.2, 0.2, 0.2, 0.3, 0.3, 0.5, 0.2, 0.2, 0.3, 0.1)	(1,0)	([-0.25 ; -0.5], [-0.5 ; 0])

On représente ci-dessous leur aspect avant usage de l'optimiseur :

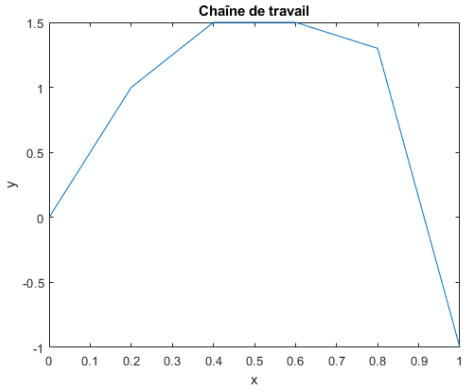


FIGURE 27 – Test 4a

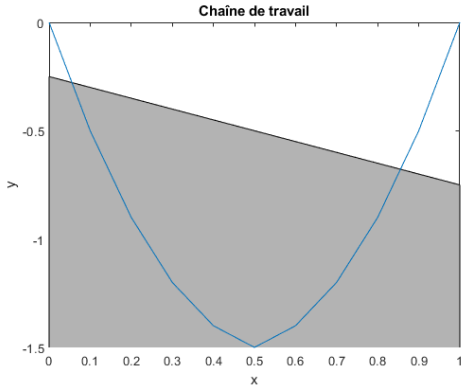


FIGURE 28 – Test 4b

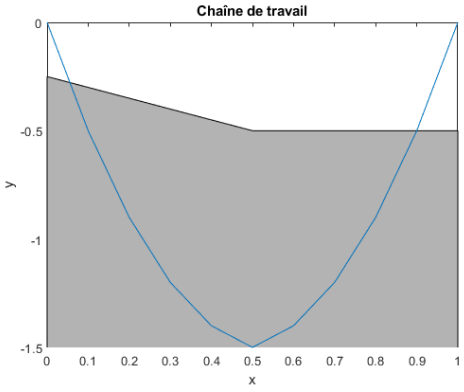


FIGURE 29 – Test 4c

FIGURE 30 – Chaînes de travail initiales $4a, 4b, 4c$

Après application dans chacun des cas de l'optimiseur initialisé avec son xy_1 respectif et le λ_1 correspondant, en prenant au maximum 1000 itérations et des bornes pour les tests d'arrêts valant 10^{-6}, on comptabilise le nombre d'itérations effectuées ainsi que les valeurs de $||F(z_*)||_\infty$, $||c_E(xy_*)||_\infty$ et $||c_I(xy_*)||_\infty$ qui caractérisent la convergence. On résume l'ensemble de ces données dans ce tableau :

Test	Nombre d'itération	$\|\|F(z_*)\|\|_\infty$	$\|\|c_E(xy_*)\|\|_\infty$	$\|\|c_I(xy_*)\|\|_\infty$
$4a$	20	$1.0019e-07$	$3.6366e-09$	pas de contrainte
$4b$	30	0.4729	$3.9441e-14$	0.6722
$4c$	43	0.4000	$1.1331e-14$	0.6590

L'aspect des chaînes en sortie est quant-à lui affiché ci-dessous :

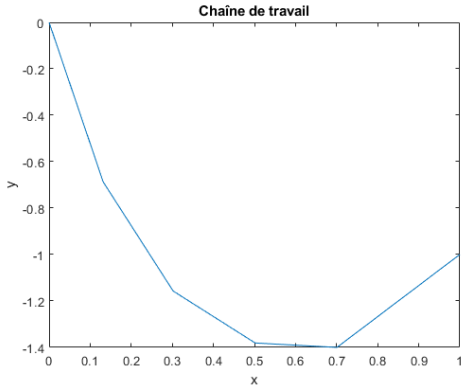


FIGURE 31 – Test 4a

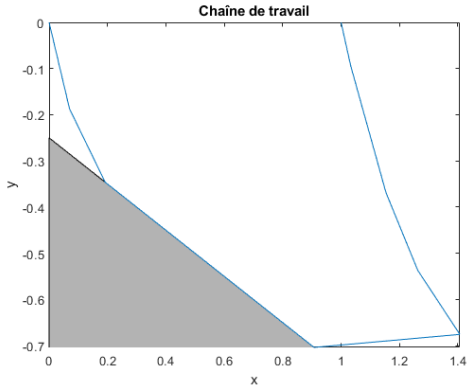


FIGURE 32 – Test 4b

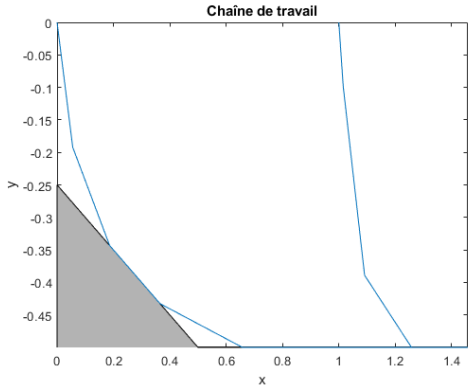


FIGURE 33 – Test 4c

FIGURE 34 – Chaînes de travail optimisées $4a, 4b$ et $4c$

On remarque alors bien dans chacun des cas une convergence des itérés vers une solution du problème des points stationnaires. Pour le cas 4a, ayant une absence de contraintes d'inégalité (pas de plancher), on remarque bien que le terme $||F(z_*)||_\infty$ est proche de zéro tout comme $||c_E(xy_*)||_\infty$ ce qui montre effectivement la convergence des itérés vers un point fixe dont l'aspect ressemble par ailleur à celui d'un minimum mais notre sens physique ne permet pas de conclure sur l'identité réelle de cette solution. Cependant, dans les deux cas suivant, $||F(z_*)||_\infty$ n'est plus proche de zéro ce qui signifie que les solutions trouvées ne sont pas des points stationnaires de l'énergie potentielle e. Ceci est en réalité cohérent avec l'aspect visuel : la solution atteinte est vraisemblablement un minimum dans l'espace admissible au-dessus du plancher pour laquelle certaines composantes de c_I sont actives.

Elles correspondent en effet à des valeurs nulles dans le vecteur $c_I(x^*)$ dont on affiche les composantes pour chaque cas test ci-dessous :

```
ci =

   -0.0977
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.2778
   -0.3444
   -0.4580
   -0.6722
```

FIGURE 35 – Test 4b

```
ci =

   -0.0856
   -0.0000
   -0.0000
   -0.0789
   -0.2289
   -0.4789
   -0.3789
   -0.4066
   -0.6590
   -0.3079
   -0.1567
   -0.0673
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.1110
   -0.4012
```

FIGURE 36 – Test 4c

FIGURE 37 – Vecteurs $c_I(x^*)$ pour les cas tests 4*b* et 4*c*

Ainsi, ce résultat est cohérent avec l'intuition, les chaînes chuteraient vers le bas pour minimiser leur énergie potentielle et atteindre son minimum (qui sera conditionnée par notre premier jeu de contraintes d'égalité sur les longueurs c_E) jusqu'à être bloquées par le plancher (qui serait notre deuxième jeu de contraintes d'inégalités sur les positions des noeuds). Néanmoins, même si cette intuition est probablement fondée, aucun argument mathématique simple peut nous permettre d'identifier si les solutions affichées sont bel et bien des minimums globaux de notre problème.

6 Version quasi-newtonienne (TP6)

6.1 Modification de l'algorithme

On peut améliorer notre algorithme OQS en évitant d'avoir à utiliser le hessien du lagrangien dans les calculs de la matrice M_k pour simplifier grandement les calculs. On parle alors de version quasi-Newtonnienne de l'algorithme. La matrice

M_k générée ne sera donc plus qu'une approximation symétrique définie positive du hessien du lagrangien. L'algorithme générera à présent une suite primale-duale (x_k, λ_k) et la suite (M_k) de matrices symétriques positives en calculant à chaque étape la solution primale-duale (d_k, λ^{PQ}) du problème quadratique osculateur dans lequel M_k joues le rôle du hessien du lagragien. Cependant, la matrice M_{k+1} sera cette fois calculée par la formule BFGS combinée avec la correction de Powell :

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

Les différentes données sont décrites dans l'énoncé et permettent l'implémentation de l'algorithme sous Matlab et la modification du fichier sqp.m fourni avec ce rapport.

6.2 Vitesse de convergence de l'algorithme

Il semble pertinent de chercher à étudier une nouvelle fois les vitesses de convergence des algorithmes implémentés. On rappelle que cela équivaut à l'étude de la convergence du quotient :

$$\frac{||z_{k+1} - z_*||_\infty}{||z_k - z_*||_\infty^\alpha}$$

Cependant, contrairement au cas du problème P_E, l'apport de contraintes d'inégalités rend obsolète la fonction F définie précédemment, le problème P_{EI} ne pouvant plus être résolu par la recherche d'un point stationnaire annulant F. Ainsi, il nous faut trouver une nouvelle manière de déterminer la vitesse de convergence de la suite des itérés $z_k := (x_k, \lambda_k)$ vers une solution $z_* := (x_*, \lambda_*)$ sans connaître cette dernière. Ceci nous permettera d'éviter de résoudre deux fois le problème : une fois pour déterminer une approximation de z_* et une fois pour calculer le rapport $\frac{||z_{k+1}-z_*||_\infty}{||z_k-z_*||_\infty^\alpha}$ ou alors de mémoriser l'ensemble des valeurs des itérés.

Notre but est essentiellement d'établir un critère permettant d'affirmer ou de réfuter expérimentalement une convergence superlinéaire et une convergence quadratique de l'algorithme. Ceci est redu possible par l'étude de la suite des déplacements (s_k) définie par :

$$s_k := z_{k+1} - z_k$$

En effet, dans le cas de la convergence superlinéaire, on peut prouver que :

1. Si (z_k) converge superlinéairement vers z_*, alors la suite (s_k) converge superlinéairement vers 0.
2. Inversement, si (s_k) converge superlinéairement vers 0, alors (z_k) est une suite de Cauchy qui converge superlinéairement vers sa limite (dans un espace de Banach).

En effet, pour le point 1., en supposant que (z_k) converge superlinéairement vers z_*, on a que :

$$s_k = (z_{k+1} - z_*) - (z_k - z_*) = -(z_k - z_*) + o(||z_k - z_*||)$$

Ainsi, on en déduit directement l'équivalence : $s_k \sim (z_k - z_*)$. (On notera ici $x_k \sim y_k$ pour signifier l'existence de $C \geq 1$ telle qe $\forall k, C^{-1}||x_k|| \leq ||y_k|| \leq C||x_k||$).

On obtient donc de $z_{k+1} - z_* = o(||z_k - z_*||)$ que $s_{k+1} = o(||s_k||)$ ce qui correspond effectivement à la convergence superlinéaire se (s_k) vers 0.

Pour le point 2., on pose une suite (z_k) et on note (s_k) la suite des déplacements correspondants qui sera supposée converger superlinéairement vers 0. On pourra aussi supposer dans la suite que $s_k \neq 0$ pour tout $k \geq 1$ car si $s_k = 0$ avec un k assez grand, la convergence superlinéaire de (s_k) implique que $s_i = 0$ pour tout $i \geq k$ et donc que la suite (z_k) est stationnaire. Dans ce cas, elle est bien une suite de Cauchy qui converge superlinéairement vers sa limite.

Par la convergence superlinéaire de (s_k) vers 0 et par notre intêret au comportement aymptotique des suites, on pourra supposer que pour tout $k \geq 1$, on a :

$$\varepsilon_k := \sup_{i \geq k} \frac{||s_{i+1}||}{||s_i||} < 1 \tag{1}$$

Montrons que la suite (ε_k) tend vers 0 en décroissant. On pourra commencer par monter que :

$$\forall k \geq 1, \forall i \geq 1, ||s_{k+i}|| \leq \varepsilon_k^i ||s_k|| \tag{2}$$

Pour ce faire, fixons $k \geq 1$ et montrons l'inégalité par récurrence sur i. Par (1), l'inégalité (2) à lieue pour $i = 1$. Supposons qu'elle ait lieu pour un $i \geq 1$ et montrons qu'elle reste vérifiée pour $i + 1$. On sait que par (1), la décroissance de (ε_k) et l'hypothèse de récurrence, nous pouvons déduire que :

$$||s_{k+i+1}|| \leq \varepsilon_{k+i}||s_{k+i}|| \leq \varepsilon_k||s_{k+i}|| \leq \varepsilon_k^{i+1}||s_k||$$

Par récurrence, la propriété (2) est démontrée.
Montrons à présent que (z_k) est une suite de Cauchy qui converge vers une limite

z_* et que :

$$\forall \varepsilon > 0, \exists k_\varepsilon, \forall k \geq k_\varepsilon, ||z_k - z_*|| \leq (1+\varepsilon)||s_k|| \tag{3}$$

Soit $\varepsilon > 0$, pour $l > k$ et k suffisamment grand pour avoir $\frac{1}{1-\varepsilon_k} \leq 1 + \varepsilon$ on a :

$$\begin{aligned}
||z_k - z_l|| &= || \sum_{i=0}^{l-l-1} s_{k+i}|| \\
&\leq \sum_{i=0}^{l-l-1} ||s_{k+i}|| \\
&\leq (\sum_{i=0}^{l-l-1} \varepsilon_k^i)||s_k|| \\
&\leq \frac{1}{1-\varepsilon_k}||s_k|| \text{ car } \varepsilon_k < 1 \\
&\leq (1+\varepsilon)||s_k|| \text{ car } \frac{1}{1-\varepsilon_k} \leq 1+\varepsilon
\end{aligned}$$

Ainsi, (z_k) est bien une suite de Cauchy dont la limite est z_*. L'inégalité (3) est alors obtenue en passant à la limite $l \to +\infty$ dans l'inégalité ci-dessus.

Montrons maintenant que la convergence de (z_k) est superlinéaire. Pour ce faire, montrons que :

$$\forall \varepsilon > 0, \exists k_\varepsilon, \forall k \geq k_\varepsilon, ||z_k - z_*|| \geq (1-\varepsilon)||s_k|| \tag{4}$$

On pose $\varepsilon > 0$ et pour $l > k$ et k suffisamment grand pour que $\frac{\varepsilon_k}{1-\varepsilon_k} \leq \varepsilon$ on a :

$$\begin{aligned}
||z_k - z_l|| &= || \sum_{i=0}^{l-k-1} s_{k+i}|| \\
&\geq ||s_k|| - || \sum_{i=1}^{l-k-1} ||s_{k+i}|| \\
&\geq (1 - \sum_{i=1}^{l-k-1} \varepsilon_k^i)||s_k|| \\
&\geq (1 - \frac{\varepsilon_k}{1-\varepsilon_k})||s_k|| \text{ car } \varepsilon_k < 1 \\
&\geq (1-\varepsilon)||s_k|| \text{ car } \frac{\varepsilon_k}{1-\varepsilon_k} \leq \varepsilon
\end{aligned}$$

En faisant tendre $l \to +\infty$, on obtient alors bien (4). Ainsi, les inégalités (3) et (4) pour $\varepsilon \in]0,1[$ montrent que $s_k \sim (z_k - z_*)$. La convergence superlinéaire de s_k

vers 0 implique donc celle de (z_k) vers z_*. Ceci démontre le point 2.

On déduit de cette propriété un test pour l'étude de la convergence superlinéaire ou non de la suite des itérés (z_k). Il faudra en effet user de l'équivalence $s_k \sim (z_k - z_*)$ vérifiée dans le cas où une des suites est de convergence superlinéaire. Ainsi, on testera la convergence superlinéaire de (z_k) par le test suivant :

- Si (s_k) a une convergence superlinéaire, alors (z_k) aussi par le point 2.
- Sinon, par contraposée du point 1., (z_k) ne converge pas superlinéairement.

On peut établir un test similaire pour la convergence quadratique en se basant sur la propriété suivante :

1. Si (z_k) converge quadratiquement vers z_*, alors la suite (s_k) converge quadratiquement vers 0.
2. Inversement, si (s_k) converge quadratiquement vers 0, alors (z_k) est une suite de Cauchy qui converge quadratiquement vers sa limite (dans un espace de Banach).

Le point 1. se démontre aisément en considérent que si (z_k) converge quadratiquement vers z_*, alors (z_k) converge superlinéairement vers z_* et par ce qui précède on a $s_k \sim (z_k - z_*)$. Ainsi, on a déduit que :

$$\frac{||s_{k+1}||}{||s_k||^2} \sim \frac{||z_{k+1} - z_*||}{||z_k - z_*||^2} \underset{k \to +\infty}{\to} 0$$

Donc le point 1. est démontré.

La démonstration du point 2 est similaire. Si (s_k) converge quadratiquement vers 0, alors (s_k) converge superlinéairement vers 0 et donc par ce qui précède (z_k) est une suite de Cauchy qui converge vers sa limite z_* et $s_k \sim (z_k - z_*)$. Ainsi, on déduit comme précédemment la convergence quadratique de (z_k) vers z_*.

Pour l'étude de la convergence quadratique de (z_k), on pourra de se référer au test suivant :

- Si (s_k) a une convergence quadratiquement, alors (z_k) aussi par le point 2.
- Sinon, par contraposée du point 1., (z_k) ne converge pas quadratiquement.

On pourra également remarquer que l'étude de la convergence de (z_k) peut être réalisée plus précisement grâce à la relation z_* et $s_k \sim (z_k - z_*)$, vérfiée si une des deux suites est de convergence superlinéaire. Or, on peut montrer que la convergence de la suite des itérés (z_k) est superlinéaire pour l'algorithme BFGS. Ainsi, on pourra bel et bien se reporter à l'étude de la convergence de (s_k) pour étudier celle de (z_k) et pour des valeurs de la puissance α supérieures à 1.

6.3 Caractère bien posé du PQO

Comme $M_k > 0$, la complexité du PQO n'est plus que polynomiale et de plus le problème est bien borné.
Toutefois, il reste une dernière problématique qui correspond au fait que le (PQO) peut ne pas être réalisable c'est à dire que les contraintes linéarisées peuvent être incompatibles et donc qu'il n'y a pas de solution (car l'ensemble admissible est réduit à l'ensemble vide).

6.4 Approximation du hessien du lagrangien par la formule BFGS

Le fait d'avoir une bonne approximation de la hessienne permet à l'algorithme d'hériter des bonnes propriétés de convergence locale de l'algorithme de Newton, tandis que le fait d'avoir une approximation définie positive permet de réduire la complexité du problème OQS qui a alors une complexité polynomiale. Ainsi, ces deux propriétés sont primordiales.
Toutefois, elles peuvent être contradictoire car la hessienne du lagrangien n'est pas toujours définie positive aussi si notre approximation est trop proche en norme matricielle de la hessienne alors elle ne sera pas définie positive. Tandis que si la distance entre notre approximation et la hessienne du lagrangien est trop élevée alors nous perdrons les qualités de convergence de l'algorithme.

6.5 Choix de γ_k dans la formule de BFGS

L'utilisation du vecteur γ_k^l est justfiée par le fait qu'on veuille que M_k approxime le hessien du lagrangien. En ce sens, en remarquant que par un développement de Taylor-Young avec reste intégral :

$$\nabla l(x_{k+1}, \lambda_{k+1}) = \nabla l(x_k + \delta_k, \lambda_{k+1}) = \nabla l(x_k, \lambda_{k+1} + \left(\int_0^1 \nabla^2 l(x_k + t\delta_k) dt \right) \delta_k$$

On en déduit que : $\gamma_k^l = \left(\int_0^1 \nabla^2 l(x_k + t\delta_k) dt \right) \delta_k$. Ainsi, il semble pertinent d'imposer à M_k de satisfaire l'équation vérifiée par la hessienne moyenne $\int_0^1 \nabla^2 l(x_k + t\delta_k) dt$

dite équation de quasi-Newton :

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

La vérification de cette relation implique alors l'égalité $\gamma_k = \gamma_k^l$ dans la formule BFGS. Cependant, l'utilisation de cette relation ne garantie pas la définie positivité des matrices M_k, car alors on devrait avoir $\gamma_k^l \delta_k > 0$ ce qui n'est pas garanti. On préférera donc trouver γ_k tel que $\gamma_k = M_{k+1}\delta_k$ de sorte que la matrice M_{k+1} soit définie positive. Ceci est rendu possible grâce à la correction de Powell consistant à prendre :

$$\gamma_k = \theta_k \gamma_k^l + (1 - \theta_k) M_k \delta_k$$

où θ_k est pris maximal dans $]0, 1]$ de sorte que $\gamma_k^\top \delta_k \geq 0.2\delta_k^\top M_k \delta_k$ ce qui amène directement à la formule fournie dans l'énoncé.

Il est décrit dans l'énoncé qu'il est nécessaire d'avoir $\gamma_k^T \delta_k > 0$ pour que la nouvelle approximation de la hessienne du lagrangien M_{k+1} soit définie positive, c'est ce que nous clarifierons ici :
Tout d'abord, on suppose par hypothèse sur l'algorithme qu'à la k-ème itération, la matrice M_k est symétrique définie positive.
Montrons que si $\gamma_k^T \delta_k > 0$ alors $M_{k+1} > 0$:
Soit $u \in \mathbb{R}^n \backslash \{0\}$,on a alors :

$$\begin{aligned} u^T M_{k+1} u &= u^T M_k u - \frac{u^T M_k \delta_k \delta_k^T M_k u}{\delta_k^T M_k \delta_k} + \frac{u^T \gamma_k \gamma_k^T u}{\gamma_k^T \delta_k} \\ &= u^T M_k u - \frac{u^T M_k \delta_k \delta_k^T M_k u}{\delta_k^T M_k \delta_k} + \frac{\langle u, \gamma_k \rangle^2}{\gamma_k^T \delta_k} \end{aligned}$$

Or, comme $M_k > 0$ on a $u^T M_k u > 0$, par ailleurs $\langle u, \gamma_k \rangle^2 > 0$ et si $\gamma_k^T \delta_k > 0$ alors $\dfrac{\langle u, \gamma_k \rangle^2}{\gamma_k^T \delta_k} > 0$, enfin on a :

$$\begin{aligned} u^T M_k \delta_k \delta_k^T M_k u &= (u^T M_k \delta_k)^2 \text{ (car } M_k \text{ est symétrique)} \\ &= (u^T \sqrt{M_k}^2 \delta_k)^2 \text{ (où } \sqrt{M_k} \text{ est l'unique racine carrée symétrique et positive de } M_k) \\ &= \langle \sqrt{M_k} u, \sqrt{M_k \delta_k} \rangle^2 \\ &\leq ||\sqrt{M_k} u||^2 ||\sqrt{M_k} \delta_k||^2 \text{ (par inégalité de Cauchy Schwarz)} \\ &\leq u^T M_k u \delta_k^T M_k \delta_k \end{aligned}$$

On peut donc en déduire que $-\frac{u^T M_k \delta_k \delta_k^T M_k u}{\delta_k^T M_k \delta_k} \geq -u^T M_k u$, ainsi on obtient finalement :

$$u^T M_{k+1} u \geq \frac{\langle u, \gamma_k \rangle^2}{\gamma_k^T \delta_k} > 0 \text{ (lorque } \gamma_k^T \delta_k > 0)$$

Ainsi, $M_{k+1} > 0$ sous l'hypothèse $\gamma_k^T \delta_k > 0$.

Montrons maintenant que si l'on prend $\gamma_k = \gamma_k^l$, alors nous n'avons pas forcément $M_{k+1} > 0$.
Supposons,

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

Or, la courbure du lagrangien peut être négative au voisinnage de l'itération donc $\nabla l(x_{k+1}, \lambda_{k+1}) < \nabla l(x_k, \lambda_{k+1})$ tandis que $x_{k+1} > x_k$, ainsi dans ce type de situation par exemple on peut remarquer que l'on aurait $\gamma_k^T \delta_k < 0$.
Montrons alors que si $\gamma_k^T \delta_k < 0$ alors on a M_{k+1} qui n'est pas définie positive :
On peut prendre $\delta_k \neq 0$ (car on a pris $x_{k+1} > x_k$ dans notre exemple), ainsi on a,

$\delta_k^T M_{k+1} \delta_k = \delta_k^T \gamma_k$ (car $\gamma_k = \gamma_k^l$ par hypothèse et d'après l'équation de Quasi-Newton)

Or, $\delta_k^T \gamma_k < 0$ d'après notre remarque précédente donc : $\delta_k^T M_{k+1} \delta_k < 0$.
Ainsi, on en déduit qu'en prenant $\gamma_k = \gamma_k^l$, on n'a pas forcément M_{k+1} définie positive.

6.6 Initialisation de M_1 par $\eta_1 I$

Nous allons ici donner un sens à l'initialisation $M_1 = \eta_1 I$. Cette initialisation correspond à une mise à l'échelle des coefficients de M_1 pour que ces coefficients se rapprochent de ceux de la hessienne du problème, d'où le choix de ce η_1 spécifique.
Aussi, nous allons développer cela plus en détail :
Tout d'abord, il faut noter qu'à la k+1 ème itération la matrice générée par la formule de BFGS (M_{k+1}) vérifie le problème d'optimisation suivant par sa construction :

$$\begin{cases} \min_M \text{"écart"}(M, M_k) \\ \gamma_k = M\delta_k \\ M = M^T \end{cases}$$

Aussi de façon similaire en prenant la deuxième itération on a M_2 qui va vérifier :

$$\begin{cases} \min\limits_{M} \text{"écart"}(M, M_1) \\ \gamma_1 = M\delta_1 \\ M = M^T \end{cases}$$

Or, on a par défaut l'initialisation $M_1 = I$, que l'on modifie ensuite avec $M1 = \eta_1 I$, aussi nous tentons ici de justifier et de donner un sens à l'usage de $\eta_1 = \frac{||\gamma_1||_2^2}{\gamma_1^T \delta_1}$.
On peut admettre que les seules initialisation intermédiaires qui sont possibles à ce stade sont de la forme $M1 = \alpha I$ où α est un scalaire choisi afin d'avoir une "mise à l'échelle du problème", ce qui permet d'être au plus près de l'approximation de la hessienne.
Aussi, le α choisi doit vérifier le problème d'optimisation suivant afin de réaliser une mise à l'échelle :

$$\begin{cases} \min\limits_{\alpha, M} ||M - \alpha I|| \\ \gamma_1 = M\delta_1 \\ M = M^T \end{cases}$$

En effet, on a ici simplement repris le problème d'optimisation écrit préalablement en fonction de M_1, en remplaçant M_1 par αI.
Par ailleurs, on peut chercher à réaliser de façon symétrique une approximation des coefficients de l'inverse de la hessienne car c'est cette dernière qui est utilisé dans l'algorithme (il suffira alors de prendre l'inverse du scalaire trouvé pour obtenir η_1 approximant les coefficients de la hessienne).
Aussi, le problème de minimisation dont notre coefficient doit être solution est désormais :

$$\begin{cases} \min\limits_{\beta, W} ||W - \beta I|| \\ W\gamma_1 = \delta_1 \\ W = W^T \end{cases}$$

où la solution (W_2,β) vérifie $W_2 = M_2^{-1}$ et $\beta = \dfrac{1}{\eta_1}$ si η_1 a été choisi de façon approprié.
Vérifions que l'affirmation précédente est vérifié, nous étudions la solution du problème suivant :

$$\begin{cases} \min\limits_{\beta, W} ||W - \beta I||^2 \\ W\gamma_1 = \delta_1 \\ W = W^T \end{cases}$$

La norme utilisée sera la norme de Frobenius, c'est à dire $||M|| = \sqrt{tr(M^T M)}$. Aussi, on a passé la norme au carré pour permettre de facilement dérivier et le problème reste équivalent.
La fonction objectif du problème de minimisation est continue (par définition de la norme) et donc fermée. Par ailleurs, les contraintes sont linéaires donc continue, or l'ensemble admissible peut dont s'écrire comme l'image réciproque d'un fermé par une application continue, ainsi l'ensemble admissible est fermé.
Enfin, la fonction objectif est coercive donc on peut en conclure par l'application du théorème de Weierstrass que le problème admet une solution.
Par ailleurs, comme nous l'avons précisé précédemment les contraintes sont linéaires en les paramètres donc les contraintes sont qualifiées. De plus, les contraintes sont différentiables et il en est de même pour la fonction objectif donc une solution (β^*,W^*) vérifiera les conditions d'optimalités de Karush-Kuhn-Tucker :
Ici, les contraintes sont particulières $W\gamma_1 = \delta_1$ correspond à $2n_n$ contraintes tandis que $W = W^T$ correspond à $2n_n \times 2n_n$ contraintes (même si certaines de ces contraintes sont redondantes). On note les multiplicateurs associés $\lambda \in \mathbb{R}^{2n_n}$ et $\Sigma \in M_{2n_n \times 2n_n}$. Aussi, le lagrangien ℓ prend la forme suivante :

$$\ell(\beta, W, \lambda, \Sigma) = ||\beta I - W||^2 + \lambda^T(W\gamma_1 - \delta_1) + \sum_{i,j} \Sigma_{i,j}(W_{i,j} - W_{j,i})$$

Aussi, en dérivant le lagrangien par rapport aux données du problème, on obtient :

$$\frac{\partial \ell}{\partial \beta} = 2Tr(\beta I - W)$$

$$\frac{\partial \ell}{\partial W_{i,j}} = (\Sigma_{i,j} - \Sigma_{j,i}) + \lambda_i \gamma_{1j} + 2W_{i,j} \text{ si } i \neq j$$

$$\frac{\partial \ell}{\partial W_{i,i}} = \lambda_i \gamma_{1i} + 2(W_{i,i} - \eta) \text{ si } i = j$$

Les conditions de Karush-Kuhn-Tucker peuvent donc se réécrire sous la forme :

$$\begin{cases} Tr(W) = 2n_n\beta \\ \Sigma - \Sigma^T + \lambda\gamma_1^T + 2(W - \beta I) = 0 \\ W = W^T \\ W\gamma_1 = \delta_1 \end{cases}$$

Puis en combinant les conditions $W^\top = W$ et $\Lambda - \Lambda^\top + \lambda\gamma_1^\top + 2(W - \beta I) = 0$, on trouve alors la condition $\frac{1}{2}(\gamma_1\lambda^\top + \lambda\gamma_1^\top) + 2(W - \beta I) = 0$ et donc les conditions de Karush-Kuhn-Tucker deviennent :

$$\begin{cases} \mathrm{Tr}(W) = 2n_n\beta \\ \frac{1}{2}(\gamma_1\lambda^\top + \lambda\gamma_1^\top) + 2(W - \beta I) = 0 \\ W^\top = W \\ W\gamma_1 = \delta_1 \end{cases}$$

Ainsi, on a ici d'après la seconde équation en multipliant par γ_1 à droite et en utilisant la relation de quasi-Newton ($W\gamma_1 = \delta_1$) que :

$$||\gamma_1||^2\lambda + 2(\delta_1 - \beta\gamma_1) = 0$$

Donc, $\lambda = \dfrac{2(\beta\gamma_1 - \delta_1)}{||\gamma_1||^2}$.
Aussi, l'une des conditions de Karush-Kuhn-Tucker (celle sur les dérivées selon les coefficients diagonaux de W), nous donne que $W_{ii} = \eta - \frac{1}{2}\gamma_{1i}\lambda_i$ donc :

$$W_{ii} = \beta - \frac{(\beta\gamma_{1i}^2 - \delta_{1i}\gamma_{1i})}{||\gamma_1||^2}$$

Ainsi, on obtient,

$$Tr(W) = 2n_n\beta - (\beta - \frac{\delta_1^T\gamma_1}{||\gamma_1||^2})$$

Par ailleurs, on a aussi $Tr(W) = 2n_n\beta$ (encore d'après les conditions (KKT)), donc on obtient grâce à l'équation précédente :

$$(\beta - \frac{\delta_1^T\gamma_1}{||\gamma_1||^2}) = 0$$

Ainsi, on trouve un coefficient optimisé égal à $\beta = \dfrac{\gamma_1^T\delta_1}{||\gamma_1||^2}$, qui permet donc une approximation des coefficients de l'inverse de la hessienne.
Ainsi, pour être utilisé dans notre problème (qui met à jour la hessienne et non son inverse), il suffira donc d'inverser ce coefficient et l'on retrouve donc :

$$\eta_1 = \frac{||\gamma_1||^2}{\gamma_1^T\delta_1}$$

6.7 Etude des cas tests

On applique par suite notre algorithme pour les cas tests $5a, 5b, 5c$ et $5d$ tels que le premier point de fixation de la chaîne est $(0, 0)$, de plus les informations relatives à chaque cas-tests sont décrites dans le tableau ci-après :

Cas test	Noeuds intermédiaires	Longueur des barres	(a,b)	(R,S)
5a	(0.2,-0.4),(0.5,-0.6),(0.8,-0.4),(1.0,-0.2),(1.2,0.)	(0.5, 0.3, 0.4, 1.2, 0.3, 0.3)	(0,0)	(-1,-0.1)
5b	(-2,1),(0,-2),	(3, 2.5, 2.5)	(0,-4)	([-6 -10],[-2 100])
5c	(0.1,-0.3),(0.2,-0.1),(0.1,0.2),(-0.1,0.4),(-0.3,0.3),(-0.2,0.2),(-0.2,0)	(0.1, 0.2, 0.3, 0.4, 0.5, 0.4, 0.3, 0.1)	(0,0)	([-1 -0.2 -1.0], [-7.0 0.0 7.0])
5d	(1,3),(2,1),(1,2),(1,4), (3,3),(2,2),(2,0)	(5, 4, 3, 2, 2, 3, 4, 5)	(4,14)	([-30 ;-12 ;-3 ;0 ;0 ;-3 ;-12 ;-30], [-10 ;-6 ;-3 ;-1 ;1 ;3 ;6 ;10]

On représente ci-dessous leur aspect avant usage de l'optimiseur :

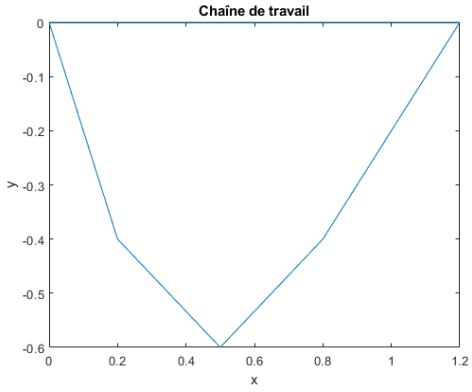


FIGURE 38 – Test 5a

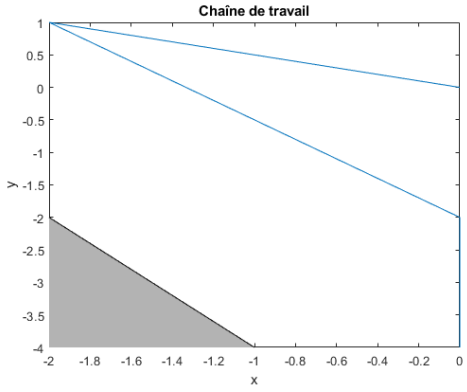


FIGURE 39 – Test 5b

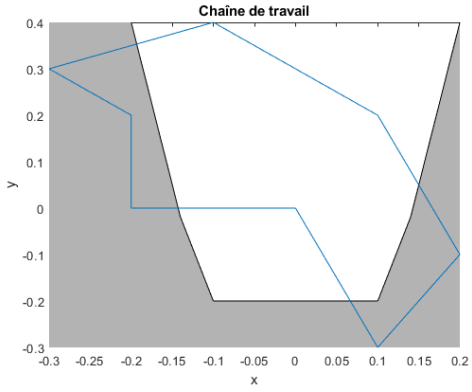


FIGURE 40 – Test 5c

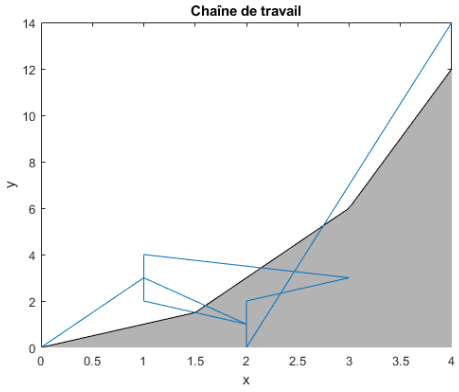


FIGURE 41 – Test 5d

FIGURE 42 – Chaînes de travail initiales $5a, 5b, 5c$ et $5d$

Après application dans chacun des cas de l'optimiseur initialisé avec son xy_1 respectif et le λ_1 correspondant, en prenant au maximum 1000 itérations et des bornes pour les tests d'arrêts valant 10^{-6}. On peut alors utiliser l'affichage spécifique des sorties de l'optimiseur pour obtenir un aperçu du déroulement de l'algorithme dans chacun des cas-tests :

Pour le cas-test 5a, on aperçoit que le convergence se fait en 21 itérations seulement, le gradient du lagrangien, les contraintes d'égalités, et le produit scalaire des contraintes d'inégalités par les multiplicateurs lagrangien tendent vers 0 rapidement, ce qui confirme la convergence. On peut voir que la correction de Powell n'est pas toujours utilisé, ce qui indique que notre matrice fourni par l'algorithme de BFGS était déjà définie positive sans correction. Par ailleurs, le conditionnement reste faible dans les cas où la correction de Powell n'a pas lieu mais on peut

iter	|gl|	|ce|	(ci,lmi)	|x|	|lm|	Powell	cond(M)
1	1.04e+02	8.08e+00	-2.72e-09	2.7e+00	1.11e+01	4.1e-02	7.5e+01
2	1.89e+00	2.17e+00	-5.36e-10	1.7e+00	3.08e-01	1.0e+00	4.8e+01
3	9.40e-01	7.97e-01	-1.29e-10	1.4e+00	9.81e-01	1.0e+00	3.8e+01
4	1.65e+00	5.20e-01	-1.86e-09	1.4e+00	2.18e+00	1.0e+00	2.1e+01
5	4.25e+00	5.35e-01	-7.41e-16	1.1e+00	1.89e+00	1.0e+00	3.7e+01
6	1.28e+00	1.62e-01	-4.40e-16	1.0e+00	1.98e+00	1.0e+00	3.2e+01
7	4.30e-01	5.17e-02	2.54e-20	1.0e+00	2.02e+00	1.0e+00	4.4e+01
8	3.20e-01	7.53e-03	2.10e-16	9.9e-01	1.89e+00	1.0e+00	9.7e+01
9	4.16e-01	1.66e-02	-1.74e-11	9.7e-01	1.84e+00	6.5e-01	1.9e+02
10	9.05e-01	6.10e-01	-2.84e-11	9.2e-01	1.45e+00	6.1e-01	6.6e+02
11	7.81e-01	4.70e-02	-1.69e-08	9.2e-01	2.05e+00	1.0e+00	4.3e+02
12	1.27e+00	1.68e-01	-8.78e-17	1.1e+00	1.82e+00	1.0e+00	3.1e+02
13	3.83e-01	3.81e-02	-1.21e-08	9.2e-01	1.70e+00	1.0e+00	2.9e+02
14	2.40e-01	1.37e-02	-2.85e-13	9.3e-01	1.64e+00	1.0e+00	1.2e+02
15	1.14e-01	1.88e-03	-3.02e-16	9.3e-01	1.68e+00	1.0e+00	1.2e+02
16	3.53e-02	2.91e-04	-5.57e-15	9.3e-01	1.66e+00	1.0e+00	1.0e+02
17	1.19e-02	6.39e-06	-3.54e-15	9.3e-01	1.64e+00	1.0e+00	6.3e+01
18	1.20e-03	1.20e-06	-1.74e-15	9.3e-01	1.63e+00	1.0e+00	5.8e+01
19	8.63e-05	4.09e-09	-1.50e-15	9.3e-01	1.63e+00	1.0e+00	5.9e+01
20	3.13e-06	3.82e-11	-2.93e-15	9.3e-01	1.63e+00	1.0e+00	6.1e+01
21	8.02e-08	1.48e-14	-2.56e-15	9.3e-01	1.63e+00	1.0e+00	6.2e+01

FIGURE 43 – Sortie de l'optimiseur (cas-test 5a)

remarquer qu'elle augmente lorsque la correction de Powell a lieu, par exemple à la 10ème itération où elle vaut 660 (environ 10 fois plus que dans les cas où il n'y a pas de correction), ce qui est logique car l'on s'éloigne alors du véritable hessien.

iter	|gl|	|ce|	(ci,lmi)	|x|	|lm|	Powell	cond(M)
1	5.32e+00	2.12e+00	-2.37e-08	3.2e+00	1.48e+00	3.4e-01	2.0e+01
2	1.64e+00	2.70e-01	-3.54e-10	3.0e+00	7.36e-01	4.3e-01	3.0e+01
3	2.54e+00	2.46e-01	3.52e-10	3.0e+00	1.24e+00	9.2e-02	4.3e+02
4	8.05e-01	5.45e-02	2.45e-09	3.0e+00	7.18e+00	1.0e+00	1.1e+02
5	6.27e-01	1.31e-02	-9.21e-12	3.0e+00	1.24e+01	1.0e+00	2.4e+02
6	5.73e-01	3.27e-03	2.25e-14	3.0e+00	2.13e+01	1.0e+00	4.9e+02
7	5.62e-01	8.17e-04	-8.81e-12	3.0e+00	3.89e+01	1.0e+00	9.7e+02
8	5.60e-01	2.04e-04	-5.22e-11	3.0e+00	7.39e+01	1.0e+00	1.9e+03
9	5.59e-01	5.10e-05	5.62e-11	3.0e+00	1.44e+02	1.0e+00	3.9e+03
10	5.59e-01	1.28e-05	-1.79e-07	3.0e+00	2.84e+02	1.0e+00	7.7e+03
11	5.62e-01	3.21e-06	-1.16e-05	3.0e+00	5.64e+02	1.0e+00	1.5e+04
12	2.19e+00	3.96e-06	-2.40e-03	3.0e+00	1.50e+03	1.0e+00	4.1e+04
13	4.33e+00	3.10e-06	-1.84e-03	3.0e+00	3.71e+03	1.0e+00	1.0e+05
14	5.10e-01	4.64e-08	1.18e-03	3.0e+00	1.58e+03	1.0e+00	4.3e+04
15	4.32e-01	5.07e-07	-2.44e-04	3.0e+00	2.13e+03	1.0e+00	5.8e+04
16	1.93e-01	1.83e-08	2.29e-04	3.0e+00	3.40e+03	1.0e+00	9.3e+04
17	3.22e-02	7.80e-09	3.41e-04	3.0e+00	3.08e+03	1.0e+00	8.4e+04
18	4.08e-03	3.86e-10	3.43e-04	3.0e+00	3.26e+03	1.0e+00	8.9e+04
19	2.66e-03	3.57e-10	3.58e-04	3.0e+00	3.14e+03	1.0e+00	8.6e+04
20	1.07e-03	1.38e-10	3.50e-04	3.0e+00	3.22e+03	1.0e+00	8.8e+04
21	4.69e-04	6.33e-11	3.56e-04	3.0e+00	3.16e+03	1.0e+00	8.7e+04
22	1.89e-04	2.48e-11	3.52e-04	3.0e+00	3.20e+03	1.0e+00	8.8e+04
23	8.04e-05	1.08e-11	3.55e-04	3.0e+00	3.18e+03	1.0e+00	8.7e+04
24	3.30e-05	4.40e-12	3.53e-04	3.0e+00	3.19e+03	1.0e+00	8.7e+04
25	1.38e-05	1.92e-12	3.54e-04	3.0e+00	3.18e+03	1.0e+00	8.7e+04
26	5.71e-06	7.48e-13	3.54e-04	3.0e+00	3.19e+03	1.0e+00	8.7e+04
27	2.38e-06	3.41e-13	3.54e-04	3.0e+00	3.18e+03	1.0e+00	8.7e+04
28	9.88e-07	1.36e-13	3.54e-04	3.0e+00	3.19e+03	1.0e+00	8.7e+04

FIGURE 44 – Sortie de l'optimiseur (cas-test 5b)

Pour le cas-test 5b, on aperçoit que le convergence se fait en 28 itérations, de même que précédemment le gradient du lagrangien, les contraintes d'égalités, et le produit scalaire des contraintes d'inégalités par les multiplicateurs lagrangien tendent vers 0 rapidement. La correction de Powell est forte aux premières itérations, ainsi l'on perd en conditionnement. Toutefois elle n'est plus utilisé dès la 4ème itération.

iter	|gl|	|ce|	(ci,lmi)	|x|	|lm|	Powell	cond(M)
1	1.46e+01	7.34e-01	-2.86e-07	1.1e+00	2.77e+01	9.7e-02	1.4e+02
2	1.17e+00	1.51e-01	-1.26e-17	7.9e-01	3.66e+00	1.0e+00	1.4e+02
3	9.98e-01	2.00e-02	-2.46e-11	6.2e-01	1.25e+00	7.2e-01	2.0e+02
4	3.03e-01	1.49e-03	-3.18e-10	5.9e-01	6.21e+00	1.0e+00	3.8e+02
5	2.46e-01	1.84e-03	-1.17e-12	6.0e-01	8.18e+00	1.0e+00	4.2e+02
6	1.35e-01	1.54e-03	-9.47e-18	6.0e-01	7.22e+00	1.0e+00	4.1e+02
7	2.19e-02	1.83e-05	-8.03e-18	5.9e-01	7.50e+00	1.0e+00	5.7e+02
8	2.15e-03	5.65e-08	-1.98e-09	5.9e-01	7.48e+00	1.0e+00	4.5e+02
9	4.17e-04	3.42e-08	-5.35e-17	5.9e-01	7.48e+00	1.0e+00	3.7e+02
10	1.18e-06	4.39e-11	-3.13e-09	5.9e-01	7.48e+00	1.0e+00	3.7e+02

FIGURE 45 – Sortie de l'optimiseur (cas-test 5c)

Pour le cas-test 5c, on aperçoit que le convergence se fait en 10 itérations seulement, aussi les conditions initiales des noeuds ont été choisi par nous-même. Aussi, leur position devait être proche d'un point stationnaire ce qui a permis une convergence rapide. Comme précédemment, le gradient du lagrangien, les contraintes d'égalités, et le produit scalaire des contraintes d'inégalités par les multiplicateurs lagrangien tendent vers 0 rapidement, ce qui atteste d'une convergence rapide dont nous étudierons la nature plus précisément dans la partie suivante.

iter	|gl|	|ce|	(ci,lmi)	|x|	|lm|	Powell	cond(M)
1	1.73e+02	5.62e+01	-2.59e-09	1.1e+01	1.03e+01	1.5e-01	2.3e+02
2	3.99e+01	1.85e+01	-4.66e-10	9.3e+00	3.29e+00	1.0e+00	2.0e+02
3	9.27e+00	6.50e+00	-2.58e-12	9.0e+00	2.08e+00	1.0e+00	2.1e+02
4	8.89e+00	2.48e+00	-8.86e-11	9.2e+00	5.52e+00	1.0e+00	1.3e+02
5	7.04e+00	2.52e+00	-4.22e-11	9.3e+00	2.30e+00	1.0e+00	1.3e+02
6	7.46e+00	9.90e-01	-2.62e-15	9.5e+00	2.35e+00	8.8e-01	1.5e+02
7	1.00e+01	4.00e-01	-8.04e-15	9.5e+00	3.92e+00	4.4e-01	2.2e+02
8	6.86e+00	1.11e+00	-1.71e-10	9.4e+00	6.64e+00	1.0e+00	2.0e+02
9	6.55e+00	3.54e+00	-2.04e-09	9.1e+00	6.66e+00	1.0e+00	1.7e+02
10	2.54e+00	2.85e-01	-5.81e-16	9.1e+00	4.64e+00	1.0e+00	1.6e+02
11	1.17e+00	1.08e-02	-4.88e-13	9.1e+00	4.00e+00	1.0e+00	1.4e+02
12	9.74e-01	3.42e-02	-3.63e-09	9.1e+00	4.46e+00	1.0e+00	1.5e+02
13	7.62e-01	5.52e-02	2.81e-15	9.0e+00	4.69e+00	1.0e+00	1.5e+02
14	7.06e-01	6.36e-03	2.83e-15	9.0e+00	4.81e+00	1.0e+00	8.3e+01
15	3.93e-01	1.97e-02	4.86e-16	9.0e+00	4.68e+00	1.0e+00	7.5e+01
16	1.71e-01	3.96e-03	-1.09e-13	9.0e+00	4.77e+00	1.0e+00	7.6e+01
17	4.96e-02	1.78e-03	-7.98e-14	9.0e+00	4.73e+00	1.0e+00	7.6e+01
18	7.92e-03	8.65e-05	1.34e-15	9.0e+00	4.75e+00	1.0e+00	7.5e+01
19	8.14e-04	1.08e-06	1.74e-15	9.0e+00	4.75e+00	1.0e+00	7.4e+01
20	6.32e-05	1.31e-08	2.97e-15	9.0e+00	4.75e+00	1.0e+00	7.4e+01
21	4.38e-06	8.56e-11	-1.26e-13	9.0e+00	4.75e+00	1.0e+00	7.3e+01
22	2.84e-07	5.47e-13	-9.66e-09	9.0e+00	4.75e+00	1.0e+00	7.1e+01

FIGURE 46 – Sortie de l'optimiseur (cas-test 5d)

Enfin, pour le dernier cas-test que nous avions créé et qui donnait l'occasion de suivre le comportement de la chaine en présence d'un plancher approximant un

demi-cercle. On peut voir qu'il n'y a pas de problème de convergence vers un point stationnaire, en effet l'algorithme converge en 22 itérations et les conditions de (KKT) tendent vers 0 comme dans les autres cas.
On remarque alors dans tous les cas une convergence rapide de l'algorithme en un nombre réduit d'itérations. Le gradient du lagrangien ainsi qui les contraintes d'inégalités semblent tendre vers 0 tandis que les valeurs de `(ci,lmi` restent très petites. Le conditionnement de la matrice M_k reste quant à lui borné ce qui nous garanti une certaine stabilité de notre méthode vis à vis des erreurs sur les solutions impliquées par des erreurs sur les données.

L'aspect des chaînes en sortie est quant-à lui affiché ci-dessous :

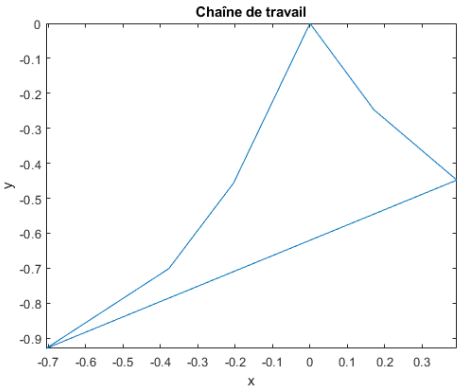


FIGURE 47 – Test 5a

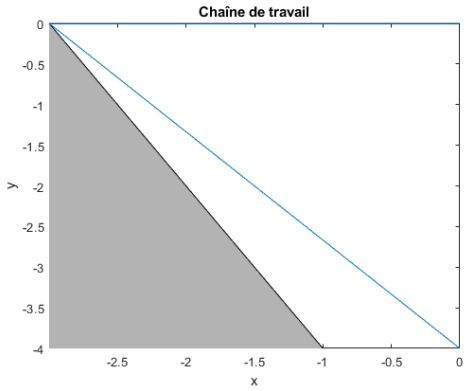


FIGURE 48 – Test 5b

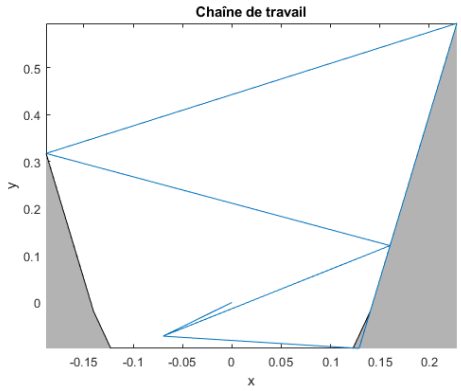


FIGURE 49 – Test 5c

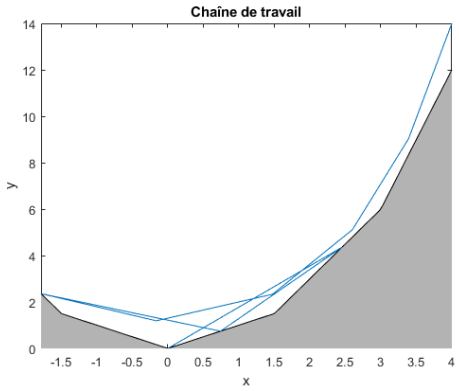


FIGURE 50 – Test 5d

FIGURE 51 – Chaînes de travail optimisées $5a, 5b, 5c$ et $5d$

6.8 Étude de la convergence (cas-tests TP4 et TP5)

Par ailleurs, on peut aussi étudier plus en détail le type de convergence (quadratique, super-linéaire, linéaire...) de la suite des itérés générées par l'algorithme dans chacun des cas-tests. C'est ce que nous allons donc faire par la suite :

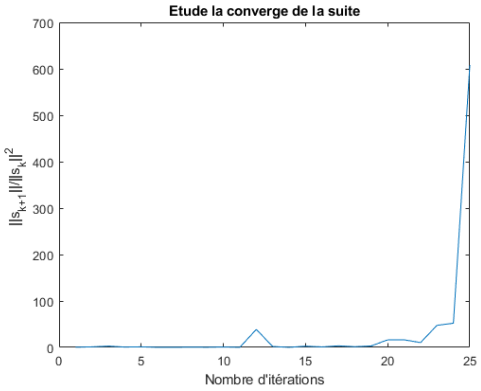


FIGURE 52 – Test 4a

FIGURE 53 – Test 4b

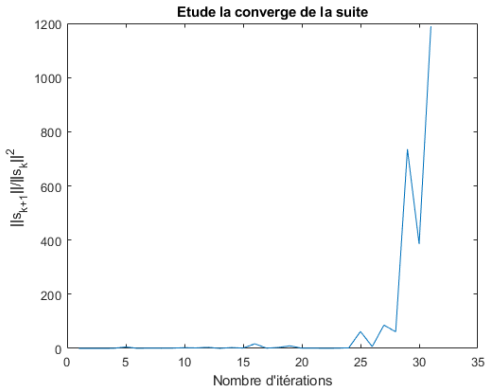


FIGURE 54 – Test 4c

FIGURE 55 – Étude de la convergence quadratique pour les cas-tests 4a,4b et 4c

Dans les cas-tests 4a, 4b et 4c, les courbes nous indiquent clairement que le quotient : $\frac{||s_{k+1}||}{||s_k||^2}$ ne tend pas vers 0, en effet le quotient semble même diverger. Ainsi, il n'y a pas convergence quadratique de la suite des itérés. Ceci correspond aux résultats théorique car l'on perd la propriété nécessaire de convergence quadratique dans le cas des algorithmes de Quasi-Newton. Aussi, pour ces algorithmes d'après nos remarques théoriques précédentes on devrait toutefois conserver la convergence

super-linéaire, vérifions cela avec le critère portant sur $\frac{||s_{k+1}||}{||s_k||}$:

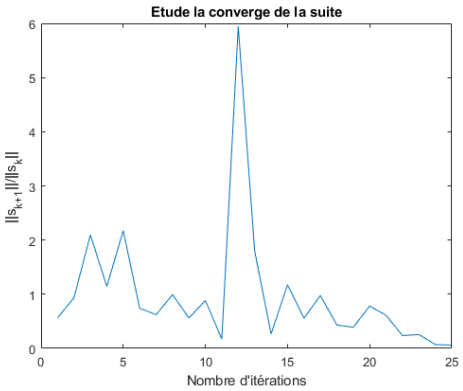


FIGURE 56 – Test 4a

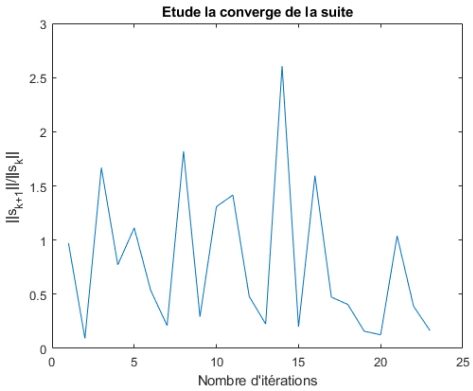


FIGURE 57 – Test 4b

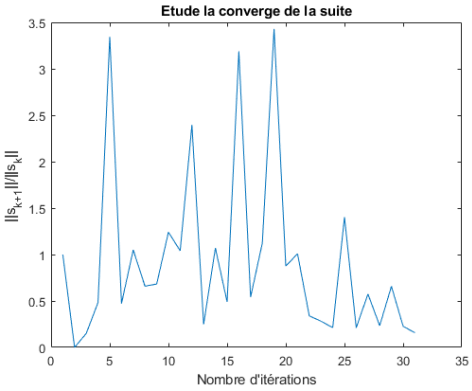


FIGURE 58 – Test 4c

FIGURE 59 – Étude de la convergence super-linéaire pour les cas-tests 4a,4b et 4c

On peut alors remarquer que cette fois le quotient $\frac{||s_{k+1}||}{||s_k||}$ semble tendre vers 0, ce qui est un critère prouvant la convergence super-linéaire de la suite des itérés généré par notre algorithme.

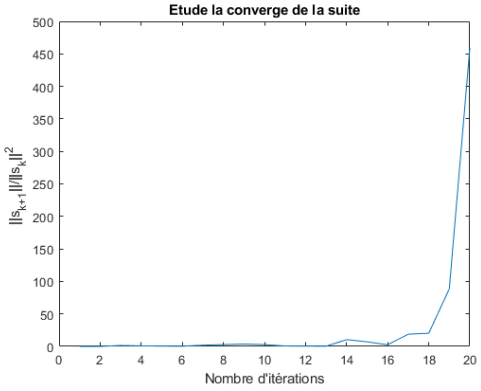


FIGURE 60 – Test 5a

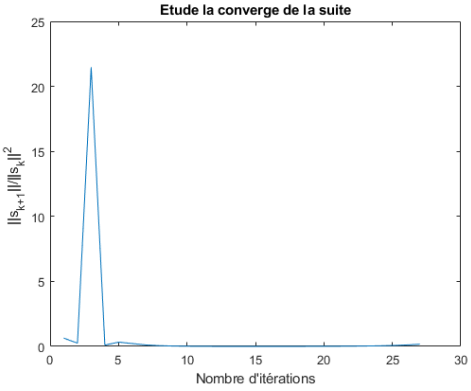


FIGURE 61 – Test 5b

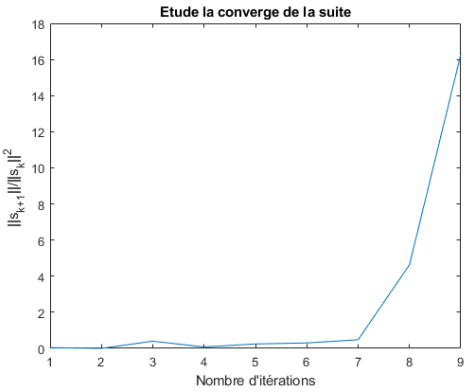


FIGURE 62 – Test 5c

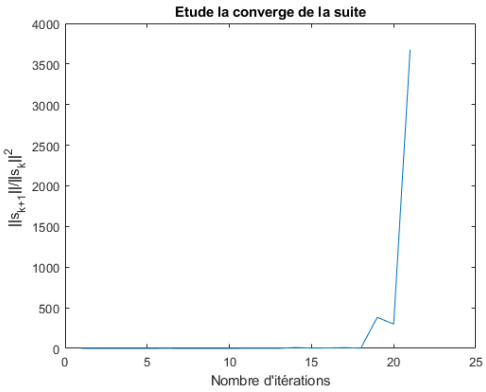


FIGURE 63 – Test 5d

FIGURE 64 – Étude de la convergence quadratique pour les cas-tests $5a, 5b, 5c$ et $5d$

On retrouve dans les cas-tests 5a, 5c et 5d, des courbes qui comme dans les cas-tests 4a,4b et 4c nous indiquent clairement que le quotient : $\frac{||s_{k+1}||}{||s_k||^2}$ ne tend pas vers 0, en effet le quotient semble même diverger. Ainsi, il n'y a pas convergence quadratique de la suite des itérés. Ceci correspond aux résultats théorique car l'on perd la propriété nécessaire de convergence quadratique dans le cas des algorithmes de Quasi-Newton. Aussi, pour ces algorithmes d'après nos remarques théoriques précédentes on devrait toutefois conserver la convergence super-linéaire, vérifions cela avec le critère portant sur $\frac{||s_{k+1}||}{||s_k||}$. Par ailleurs, on peut aussi remarquer que dans le cas-test 5b, la suite des itérés générés par le programme converge quadratiquement, ce qui est plutôt exceptionnel dans le cas des algorithmes de quasi-Newton (même si cela reste possible comme on le voie ici).

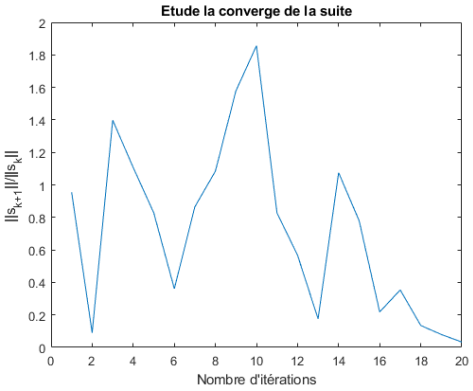


FIGURE 65 – Test 5a

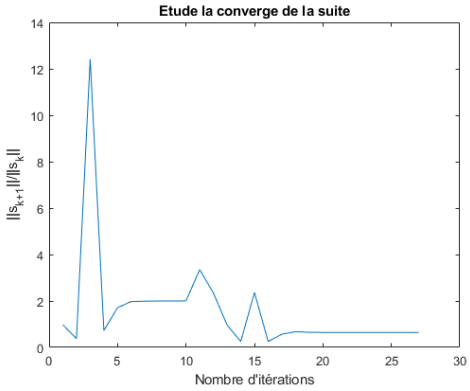


FIGURE 66 – Test 5b

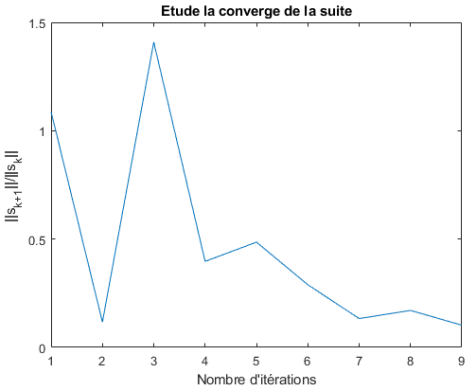


FIGURE 67 – Test 5c

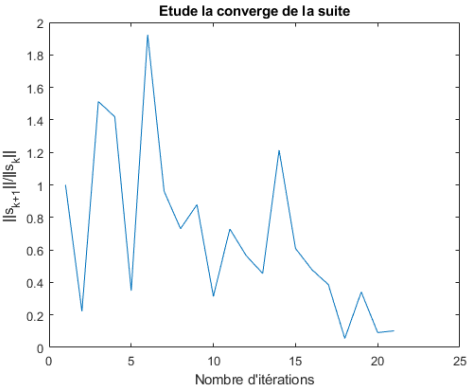


FIGURE 68 – Test 5d

FIGURE 69 – Étude de la convergence super-linéaire pour les cas-tests $5a, 5b, 5c$ et $5d$

On peut alors remarquer que comme pour les cas-test n°4 le quotient $\frac{||s_{k+1}||}{||s_k||}$ semble tendre vers 0, ce qui est un critère prouvant la convergence super-linéaire de la suite des itérés générés par notre algorithme dans chacun des cas-tests n°5.

7 Conclusion

Dans ce TP, nous avons donc pu dans un premier temps initialiser les données du problème, programmer un simulateur de notre problème et vérifier certaines de ces fonctionnalités manuellement.

Dans un second temps, nous avons programmé un optimiseur utilisant l'algorithme de Newton permettant l'approximation de points stationnaires d'une fonction et nous avons pu l'appliquer, en combinaison avec notre simulateur, à notre cas de minimisation du potentiel de la chaîne. Cela aura été aussi l'occasion de mettre en avant la vitesse de convergence quadratique de l'algorithme de Newton dans un cas pratique.

Ensuite, dans un troisième et dernier temps, nous avons pu mettre en place un technique de globalisation par recherche linéaire afin de forcer la convergence de notre algorithme de Newton dans plus de cas d'utilisation, au prix de l'imbrication d'une nouvelle boucle et d'une nouvelle suite d'itérés α_k.

Cependant, face aux difficultés implquées par la prise en considération d'un sol, nous avons du changer nos algorithme et remplacer l'algorithme de Newton par un algorithme OQS, perdant alors la convergence quadratique mais au profit d'une résolution d'un problème "mieux" posé et d'une facilité de calcul (on ne calcule plus le hessien du lagrangien).

Ce TP a mis en avant la non-trivialité de la résolution d'un problème de minimisation avec contraintes d'égalité puis d'inégalité en montrant la complexité de la résolution informatique des points stationnaires qui est un problème prérecquis au problème initial de minimisation. L'algorithme implémenté reste donc à améliorer afin d'aller au-delà de la recherche des points stationnaires pour se pencher sur la recherche de la solution à notre problème de minimisation.