Ecole Nationale Supérieure
des Techniques Avancées

Rapport de Projet

OPT202

Emma de Charry et Lucien Perdrix

# Numerical Project in Python n.1

An articulated chain

## 1 Optimality conditions

### 1.1 Question 1

The problem is not convex. The cost function is indeed convex, but the admissible set is not. This admissible set is defined by c(z) = 0. For a = 0 and b = L, as an example, the solution being of the form z = (x,y), the solution z = (x,y) also belongs to the admissible set.
Indeed, we would like to recall that c(x,y) = c(x,-y). The mean vector of these two admissible solutions is z = (x,0), which is not admissible : it corresponds to the case where the bar considered would be horizontal and attached between the points (x, 0) and (x, L/2). As the bars have a length of L, this case is impossible. Thus, the admissible set is not convex, and so the problem is not either.

### 1.2 Question 2

In this question we are going to treat the qualification of the constraints. In order for the constraints to be qualified, a necessary condition is to say that the Jacobian matrix of the constrains is surjective. In our case the Jacobian matrix of the constraints c is the transpose of the matrix $\nabla^T c(z)$ : This is the matrix below :

$$\nabla_z c(z) = 2 \left[\begin{array}{cccc|cccc} x_1 & 0 & \cdots & 0 & y_1 & 0 & \cdots & 0 \\ -(x_2-x_1) & (x_2-x_1) & \ddots & \vdots & -(y_2-y_1) & (y_2-y_1) & \ddots & \vdots \\ 0 & -(x_3-x_2) & \ddots & 0 & 0 & -(y_3-y_2) & \ddots & 0 \\ \vdots & \vdots & \ddots & (x_N-x_{N-1}) & \vdots & \vdots & \ddots & (y_N-y_{N-1}) \\ 0 & 0 & \cdots & -(x_N-a) & 0 & 0 & \cdots & -(y_N-a) \end{array}\right]$$

$\nabla_z c(z) \in \mathbb{R}^{(N+1)\times 2N}$

Let us show that this matrix is surjective. This also would mean that we can extract N + 1 independent columns from $\nabla_z c(z)$. We will proceed here by strong recurrence.

- We first select columns 1 and N+2, of which at least one must be nonzero due to the constraint $(x1, y1) \neq (0, 0)$ $(L \neq 0)$. We name this non-zero column as $e_1$.
- Then, we examine columns $e_i$, where $i \in [|1, N|]$, assuming their independence from each other. Next, we consider columns i+1 and N+i+2, of which one will also be nonzero, as $(x_i, y_i) \neq (x_i + 1, y_i + 1)$. Since this column has an extra zero at the top, it cannot be expressed as a nontrivial linear combination of the others, making it independent. This column is denoted as $e_i + 1$.
- We repeat this process for all i up to N+1, which results in N+1 independent columns and establishes the surjectivity of the matrix.

Hence, the constraints are well-qualified.

### 1.3 Question 3

Not all combinations of (a, b, L, N) yield a solution since the distance between points (0,0) and (a,b), which is $\sqrt{a^2+b^2}$, cannot exceed the length of the chain, (N+1)L.
Therefore, a necessary condition for the solution to exist is that :

$$\sqrt{a^2+b^2} \leq ((N+1)L)^2 \tag{1}$$

## 2 The Newton's Method

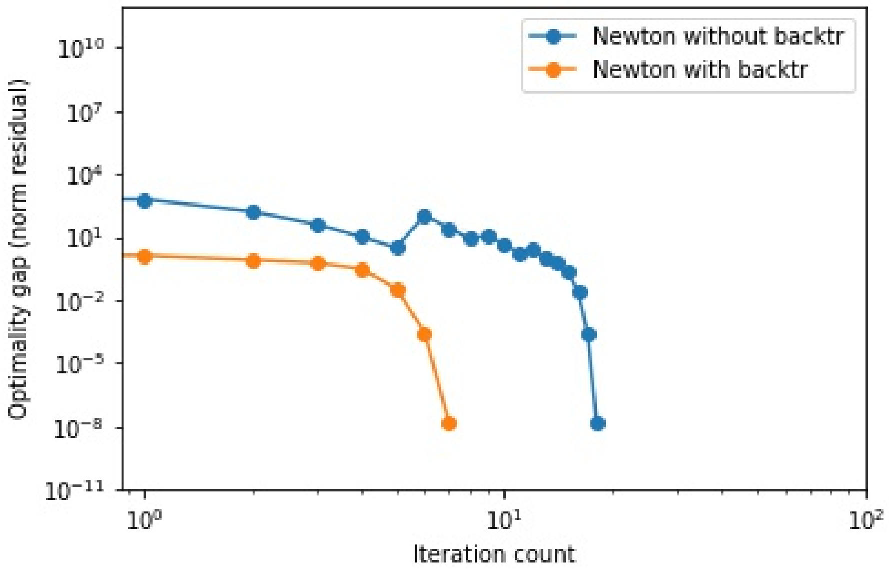
### 2.1 Question 1

We want to solve this KKT optimality conditions problem with the Newton's method. We consider the following optimality conditions

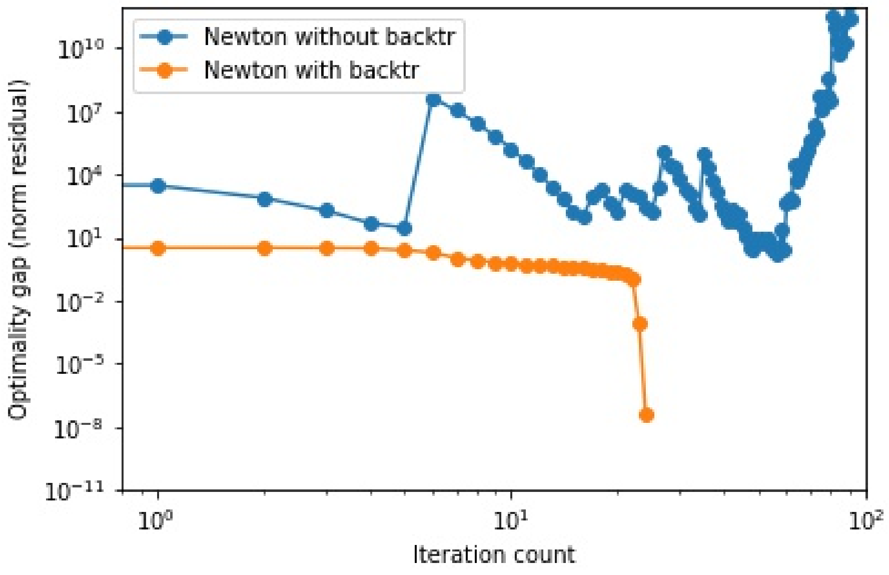
$$\begin{cases} e + \nabla_z^T c(z)\lambda = 0 \\ c(z) = 0 \end{cases} \tag{2}$$

To solve for $z$ and $\lambda$ using the Newton method, we implement the method in Python and use it to find the stationary points of (P), which we then plot.

The results demonstrate that incorporating a backtracking strategy vastly improves the convergence rate of the algorithm. The figures below illustrate that after 100 iterations, the results obtained with a backtracking strategy are much more satisfactory as a solution to the physical problem than the results obtained without backtracking with the same number of iterations.



(a) N=5, L=0.25, a=1, b=-0.1



(b) N=10, L=0.11, a=1, b=-0.4

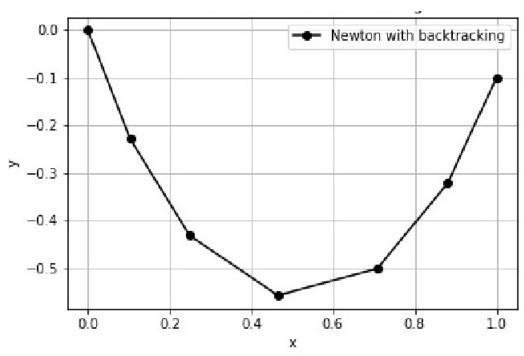
In the first case we see that for small dimension problem, backtracking strategy improves the convergence's speed bud the second case shows that for higher dimension problem we get a convergence with the backtracking strategy but not without it.

## 2.2 Question 2

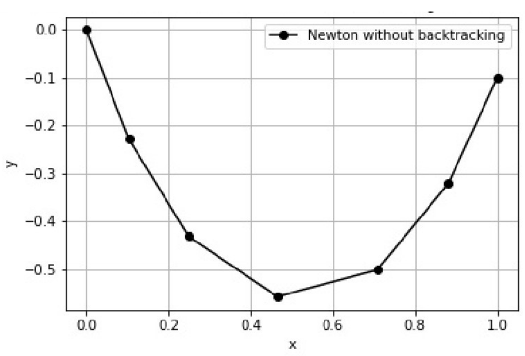
We are setting the parameters and initial conditions to :

```
N = 5
L = 0.25
a,b = 1.,-.1
x = np.zeros(2*N)
x[0:N] = np.linspace(1./(N+1), a-1./(N+1), N)
x[N:2*N] =[b*i -.1*(i-a/2.)**2 for i in x[0:N]]
lmbda = 0.1*np.random.random(N+1)
```

Both methods-with and without backtracking-are converging. As we mentionned above, the backtracking method is converging faster. The figures below illustrates that we find the same optimal value for both methods.



(a) Optimal chain find with Newton method with backtracking (N=5, L=0.25, a=1, b=-0.1)



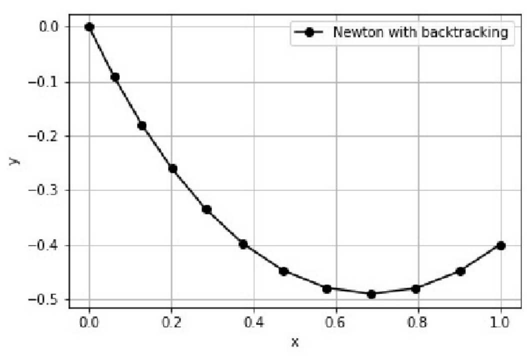
(b) Optimal chain find with Newton method without backtracking (N=5, L=0.25, a=1, b=-0.1)

If we are recompiling the algorithm for different realizations of $\lambda$ we see that the strategy without backtracking is sometimes not converging when the backtracking strategy is always converging.

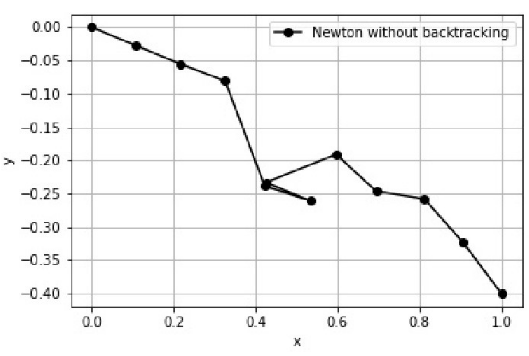
## 2.3 Question 3

Let's now turn our attention to other scenarios where N, L, and the initial conditions $\xi_0$ are different. We change the parameters to set a higher dimension problem :

```
N = 10
L = 0.11
a,b = 1.,-.4
x = np.zeros(2*N)
x[0:N] = np.linspace(1./(N+1), a-1./(N+1), N)
x[N:2*N] =[b*i -.1*(i-a/2.)**2 for i in x[0:N]]
lmbda = 0.1*np.random.random(N+1)
```



(a) Optimal chain find with Newton method with backtracking (N=10, L=0.11, a=1, b=-0.4)



(b) Optimal chain find with Newton method without backtracking (N=10, L=0.11, a=1, b=-0.4)

These figures confirm that the Newton method without backtracking is not converging in this case.

If we are recompiling the algorithm for different realization of $\lambda$ we see that the strategy without backtracking is sometimes converging but it is very rare. We conclude that the backtracking strategy is very efficient when the dimension of the problem increases.

## 3 Convex relaxation

### 3.1 Question 4

Here we are focusing on a new approach to solving the problem, by using the concept of convex relaxation. Our aim is to find a method that is less sensitive to initialization compared to the Newton method and exhibits a slower rate of divergence. To achieve this, we store our variables $(x_i, y_i)$ in a matrix called u :

$$u = [x, y] = \left[\begin{array}{c|c} x_0 & y_0 \\ x_1 & y_1 \\ \vdots & \vdots \\ x_{N+1} & y_{N+1} \end{array}\right] \in \mathbb{R}^{(N+2)\times 2}$$

We now define the following matrix X :

$$X = uu^T = \begin{bmatrix} \ddots & & & \\ & (x_i^2 - y_i^2) & x_i x_{i+1} + y_i y_{i+1} & \cdots \\ & & \ddots & \\ * & & & \ddots \end{bmatrix} \in S^{N+2}$$

### 3.2 Question 5

We will now prove that the problem $(P'')$ is convex by showing that for any $(x_1, y_1)$ and $(x_2, y_2)$ in the feasible set, $\alpha c_i(x_1, y_1) + (1-\alpha)c_i(x_2, y_2) = 0$. Letting $u = [x_1, y_1], v = [x_2, y_2], X = uu^T$, and $Y = vv^T$, we have :

$$\begin{bmatrix} I_2 & \alpha u^T + (1-\alpha)v^T \\ \alpha u + (1-\alpha)v & \alpha X + (1-\alpha)Y \end{bmatrix} \geq 0$$

If $(x_1, y_1)$ and $(x_2, y_2)$ are solutions of the optimization problem, and $m^*$ is the minimum value of the objective function, then any convex combination $(x, y) = \alpha(x_1, y_1) + (1-\alpha)(x_2, y_2)$ is also a solution because $\sum_{i=1}^{N+1} y_i = m^*$. This property is a consequence of the linearity of the objective function. Therefore, the problem $(P'')$ is convex.

### 3.3 Question 6

We can demonstrate that if $rg(X^*) = 2$, it is possible to formally consider that "$X^* = uu^T$". To prove this, we assume that there exists a solution $(u^*, X^*)$ to problem $(P'')$ for which $rg(X^*) = 2$. Additionally, we make the practically always valid assumption that $u[:, 0]$ and $u[:, 1]$ are not collinear, except in cases where the chain is perfectly taut, which are specific cases in which the admissible set is reduced to a single point. Let us first recall that $X \in S^{N+2}$ and we also have that :

$$\begin{bmatrix} I_2 & u^{*T} \\ u^* & X^* \end{bmatrix} \geq 0 \Rightarrow X^* \geq 0$$

Indeed, taking a vector

$$v^T = \begin{bmatrix} 0 & 0 & v_N \end{bmatrix} \in \mathbb{R}^{N+2}$$

and applying the definition of definite positivity, we get this result. Thus we know, as $rg(X^*) = 2$, that there exists an orthogonal matrix $Q \in O_N(\mathbb{R})$ and two strictly positive eigenvalues $\lambda_1, \lambda_2 \in \mathbb{R}_+^*$ such that we have the following :

$$X^* = Q^T \begin{bmatrix} \lambda_1 & 0 & & \cdots & 0 \\ 0 & \lambda_2 & & & 0 \\ & & 0 & & \\ \vdots & & & \ddots & \vdots \\ 0 & & \cdots & & 0 \end{bmatrix} Q$$

We can also write the following :

$$\begin{bmatrix} \lambda_1 & 0 & & \cdots & 0 \\ 0 & \lambda_2 & & & 0 \\ & & 0 & & \\ \vdots & & & \ddots & \vdots \\ 0 & & \cdots & & 0 \end{bmatrix} = WW^T = \begin{bmatrix} \sqrt{\lambda_1} & 0 \\ 0 & \sqrt{\lambda_2} \\ 0 & \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{\lambda_1} & 0 & \cdots & 0 \\ 0 & \sqrt{\lambda_2} & & 0 \end{bmatrix}$$

Let us then use here the assumption made at the beginning that the family of the two columns of u was free. Since we can complete it in a basis of $\mathbb{R}^N$ and we can do the same with the two columns of W, we can then exhibit an invertible matrix $P \in GL_N(\mathbb{R})$, which verifies :

$$Pu^*[:,0] = \begin{bmatrix} \sqrt{\lambda_1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$Pu^*[:,1] = \begin{bmatrix} 0 \\ \sqrt{\lambda_2} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

We then have :

$$\begin{bmatrix} \lambda_1 & 0 & & \cdots & 0 \\ 0 & \lambda_2 & & & 0 \\ & & 0 & & \\ \vdots & & & \ddots & \vdots \\ 0 & & \cdots & & 0 \end{bmatrix} = (Pu^*)(Pu^*)^T = P(u^*u^{*T})P^T$$

Finally we obtain that $X^* = Q^T P(u^*u^{*T})P^T Q = (Q^T P)(u^*u^{*T})(Q^T P)^T$. So since $Q^T P \in GL_N(\mathbb{R})$, the matrices $X^*$ and $u^*u^{*T}$ are congruent, which is an equivalence relation on the set of matrices. We can therefore consider in this sense that "$X^* = uu^T$".

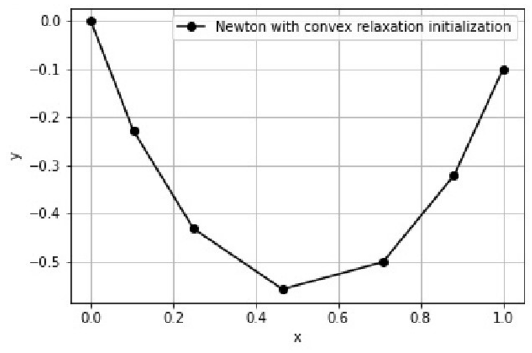
## 3.4 Question 7

Let's solve problem $(P'')$ for the first set of parameters and initial conditions(the code to solve this problem with *cvxpy* is available on our Jupyter Notebook).
The convex relaxation in this case is good. Indeed we find a solution very close to the optimal solution of $(P')$. The criteria on the eigenvalues confirms it.

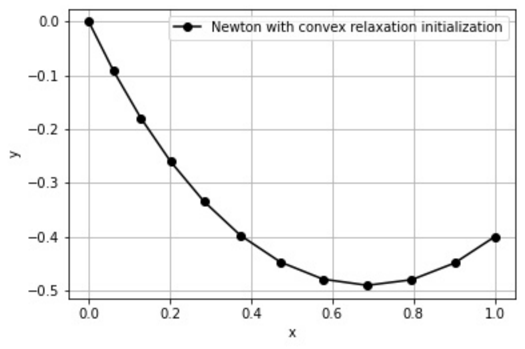
When we change the set of parameters into the $2^{nd}$ one use in this project we also find a solution respecting the eigenvalues criteria and indeed this solution is close to the optimal solution of $(P')$.

## 3.5 Question 8

Now we can use the optimal solution of $(P'')$ that we find previously to initialize the Newton method without backtracking. We get the following results :

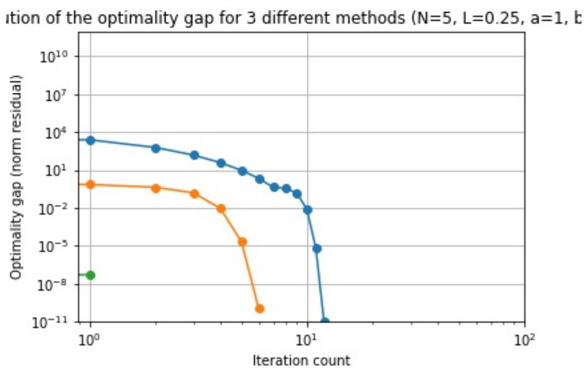


(a) Optimal chain find with Newton method with cvx relax init without backtracking (N=5, L=0.25, a=1, b=-0.1)

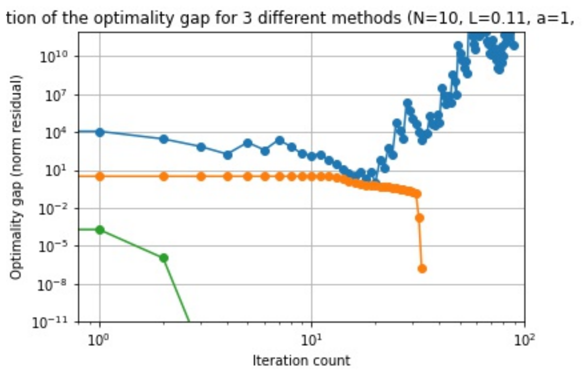


(b) Optimal chain find with Newton method with cvx relax init without backtracking (N=10, L=0.11, a=1, b=-0.4)

Let's now focus on the interesting point : the convergence speed of the method. We plot the comparison of the number of iteration needed to solve the problem for the differents methods for both set of parameters and initialization condition and we get the following figures :



(a) Comparison between number of iterations (N=5, L=0.25, a=1, b=-0.1)



(b) Comparison between number of iterations(N=10, L=0.11, a=1, b=-0.4)

We clearly see that the convex relaxation method to initialize Newton is the most efficient method even without using backtracking. It allows us to solve difficult problem with high dimension and complex parameters.