Project in TMA4180

Harlad

April 14, 2015

0.1 The problem at hand

We want to examine a hanging chain, given as equation (1),

$$E(x_{1,1}, \cdots, x_{1,n}, x_{2,1}, \cdots, y_{2,n}) := \sum_{i=1}^{n+1} m_i g \frac{x_{2,i} - x_{2,i-1}}{2} \tag{1}$$

with constraints from equation (2)

$$c_i(x_{1,1}, \cdots, x_{1,n}, x_{2,1}, \cdots, x_{2,n}) = (x_{1,i} - x_{1,i-1})^2 + (x_{2,i} - x_{2,i-1})^2 - l_i^2 = 0 \tag{2}$$

as a minimization problem, where $x_{1,i}$, $x_{2,i}$ are two spatial directions. We denote Ω as the set of feasible configurations of the chain, and $G_0 = (0,0)$, $G_1 = (a,b)$ as the start and endpoint of the chain. !!!!!!!!!!!!presentere litt Teori!!!!!!!!!!!Lage algo som lser problemet.!!!!!!!

0.2 Some theory

0.2.1 Existence of solution

First we want to show that there exists a solution. For that we get som help from theorem 0.2.1.

Theorem 0.2.1. *[1] Assume that $f : \Omega \to \mathbb{R} \cup \{+\infty\}$ Then f has at least one global minimizer in Ω if*

- *Ω is non-empty, closed and bounded.*
- *$f : \Omega \to \mathbb{R}$ continuously.*

Proof. Ω is non-empty if and only if non of the numbers $||G_0 - G_1||, l_1, \cdots, l_{n+1}$ are lagrer than the sum of all the others. Since equation (2) is closed and bounded, so must Ω be. Clearly equation (1) is linear, thus also continuous. $\square$

0.2.2 KKT conditions

KKT conditions are first order necessary conditions for a solution in nonlinear programming to be optimal [6]. It is therefore of great importance that these conditions are fulfilled. The KKT conditions with equality constraints are given in theorem 0.2.2.

Theorem 0.2.2. *[2] Assume x^* is a optimal point, and that*
$\mathcal{L}(x^*;\lambda^*) = E(x^*) - \sum_{i=1}^{n+1} \lambda_i c_i(x^*)$
If the following conditions hold, we say that x^* is a KKT point.

1. $\nabla_x \mathcal{L}(x^*, \lambda^*) = 0$
2. $c_i(x^*) = 0$

Substituting equation (1) and equation (2) into theorem 0.2.2, we obtain the KKT conditions for our problem, given in equation (3).

$$
\begin{aligned}
\nabla_{x_{1,i}} \mathcal{L}(x^*, \lambda^*) &= -2\lambda_i(x_{1,i} - x_{1,i-1}) + 2\lambda_{i+1}(x_{1,i+1} - x_{1,i}) &&= 0 \qquad (3)\\
\nabla_{x_{2,i}} \mathcal{L}(x^*, \lambda^*) &= m_i g - 2\lambda_i(x_{2,i} - x_{2,i-1}) + 2\lambda_{i+1}(x_{2,i+1} - x_{2,i}) &&= 0 \qquad (4)\\
c_i(x) &= (x_{1,i} - x_{1,i-1})^2 + (y_{2,i} - y_{2,i-1})^2 - l_i^2 &&= 0 \qquad (5)
\end{aligned}
$$

We are also interested in what will happen if $a = 0$, and there exists a feasible configuration of the chain where $x_{1,i} = 0$ for all i. This means that the whole chain is contained in a vertical line. In this special case the conditions in equation (3) simplifies a to equation (6).

$$m_i g - 2\lambda_i(x_{2,i} - x_{2,i-1}) + 2\lambda_{i+1}(x_{2,i+1} - x_{2,i}) = 0 \qquad (6)$$

$$(x_{2,i} - x_{2,i-1})^2 - l_i^2 = 0 \qquad (7)$$

More generally, equation (6) can be written as equation (8).

$$
\begin{pmatrix}
l_1 & -l_2 & 0 & \cdots & 0\\
0 & l_2 & -l_3 & \ddots & \vdots\\
\vdots & \ddots & \ddots & \ddots & 0\\
0 & \cdots & 0 & l_n & -l_{n+1}
\end{pmatrix}
\begin{pmatrix}
\lambda_1\\ \lambda_2\\ \vdots\\ \lambda_n\\ \lambda_{n+1}
\end{pmatrix}
= \frac{g}{2}
\begin{pmatrix}
m_1\\ m_2\\ \vdots\\ m_n\\ m_{n+1}
\end{pmatrix}
\qquad (8)
$$

The linear system presented in equation (8) has n equations with $n+1$ unknowns and will always yield a solution. It is important to notice that even though the KKT conditions are satisfied, this might not be the optimal solution. !!!!!!!!!!!!Skrive nr dette ikke er optimal lsning!!!!!!!!!!!!!!!!!!!!

0.2.3 LICQ conditions

LICQ conditions with equality constraints are given in definition 0.2.1.

Definition 0.2.1. *[3] Given a point x and the set of all constraints, $C(x)$, we say that the linear independence constraint qualification (LICQ) holds if the set of constraint gradients $\{\nabla c_i(x), i \in C(x)\}$ is linearly independent.*

We now want to show that if the LICQ conditions does not hold, then all the links in the chain are parallel.

Proof. Given two dependant vectors a, b, we know that

$$\lambda_1 a + \lambda_2 b = 0 \to a = -\lambda_2/\lambda_1 b$$

So they are parallel. By assumption, all $\nabla c_i(x)$ are linearly dependant thus all $\nabla c_i(x)$ are paralell. $\square$

0.3 Solving the problem

0.3.1 How to find the solution

The greates difficulty with the problem presented in equation (1) is the need to satisfy constraints in equation(2). The augmented lagrangian method is a method that simplifies the problem to only minimize equation (9) in each iteration, the full method is given in algorithm 1.

$$\mathcal{L}(x, \lambda, \mu) = E(x) - \sum_{i=1}^{n+1} \lambda c_i(x) + \frac{\mu}{2} \sum_{i=1}^{n+1} c_i(x) \tag{9}$$

Algorithm 2 is the minimizer used in algorithm (1) to minimize equation (9). It is a variant of Newton's method, which uses steepest decent method when Newtons method does not give a decent direction. The minimizer also uses a backtracking algorithm for choosing the steplength. !!!!!!!!!!!!!!!!!!Hvorfor ble denne metoden brukt!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Algorithm 1 [4]Augmented lagrangian metod

Start with x_0, λ_0 and μ_0
for $k = 1, 2, \cdots$ **do**
  Find x_{k+1} mimimizing $\mathcal{L}(x_k, \lambda_i^k, \mu_k)$
  Set $\lambda_i^{k+1} = \lambda_i^k - \mu_k c_i(x_{k+1})$
  Choose $\mu_{k+1} > \mu_k$
end for

!!!!!!!!!!!!!Harald m vise at metoden vil gi rett lsning!!!!!!!!!!!!!!!!!!!!!!

Algorithm 2 [5]Newton's method and steepest decent method with backtracking

Start with x_0, λ_0 and μ_0
Set $\gamma < 1$, $\rho < 1$
for $k = 1, 2, \cdots$ **do**
  Find $\mathcal{N} = \nabla C(x_k)$, $\mathcal{H} = \nabla^2 C(x_k)$.
  Set $p = -\mathcal{H} \backslash \mathcal{N}$
  if $\mathcal{N}^\top \mathcal{H} \mathcal{N} \geq 0$ **then**
    Set $p = -\mathcal{N}$
  end if
  Set $\alpha = 1$
  while $E(x_k + \alpha p) > E(x_k) + \gamma \alpha \mathcal{N}' p$ **do**
    $\alpha = \rho \alpha$
  end while
  $x_{k+1} = x_k + \alpha p$
end for

0.3.2 Implementation

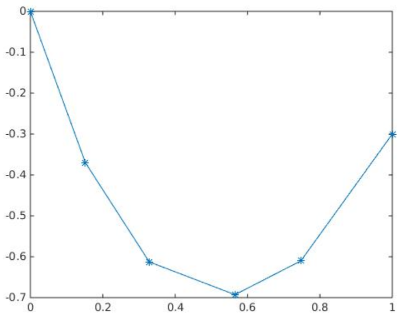
The algorithms presented in section 0.3.1 where implemented, and can be run with the script `runthing.m`, and should be self explanatory. The implementation of algorithm 1 is called `alf.m`, and the implementation of algorithm 2 is called `minimizer.m`.

0.4 Some results

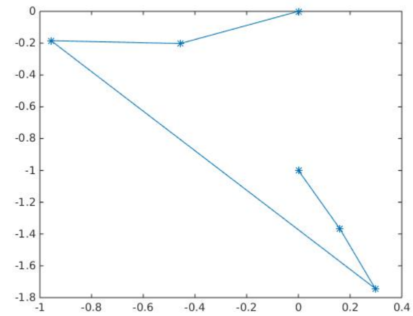
Some results from running `runthing` with different chains are shown in figure 1.

0.4.1 What about run time?

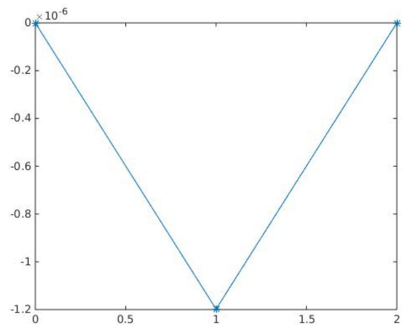
0.5 Discussion



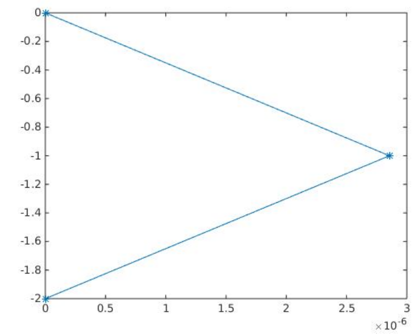
(a) $l, m = [0.4, 0.3, 0.25, 0.2, 0.4]$ with endpoint $G_1 = (1, -0.3)$.



(b) $l, m = [0.5, 0.5, 2, 0.4, 0.4]$ with endpoint $G_1 = (0, -1)$.



(c) $l, m = [1, 1]$ with endpoint $G_1 = (2, 0)$.



(d) $l, m = [1, 1]$ with endpoint $G_1 = (0, -2)$.

Figure 1: Examples of hanging chains

Bibliography

[1] J. Nocedal and S. Wright *Numerical Optimization*. Theorem 2nd edition, 2006.

[2] J. Nocedal and S. Wright *Numerical Optimization*. Theorem 12.1 page 321 2nd edition, 2006.

[3] J. Nocedal and S. Wright *Numerical Optimization*. Definition 12.4 page 321 2nd edition, 2006.

[4] J. Nocedal and S. Wright *Numerical Optimization*. Framework 17.3 page 515 2nd edition, 2006.

[5] J. Nocedal and S. Wright *Numerical Optimization*. Algorithm 3.2 page 37 2nd edition, 2006.

[6] Karush-Kuhn-Tucker conditions `http://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions`