南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

南方科技大学力学与航空航天工程系

研究报告

课程名称：高等数值分析

指导老师：单肖文

报告名称：悬链线问题的数值解法

团队成员：

姓名 彭秀枫 学号：11849095

姓名 陈泽彬 学号：11849169

姓名 尹文壮 学号：11849155

姓名 陈炫午 学号：11710323

姓名 李奕志 学号：11712206

2019 年 6 月 10 日

悬链线问题的数值解法

彭秀枫 陈泽彬 尹文壮 陈炫午 李奕志

（力学与航空航天工程系　指导教师：单肖文）

[摘要]：悬链线问题是数值分析问题中的经典问题，通过数学手段，我们可以精确描述悬链线的形状。本文通过对悬链线的代表性问题——铁链问题进行求解，得到了悬链线的基本方程，并通过控制变量法得到了不同边界条件对与悬链线形状的影响。

[关键词]：悬链线；数值分析；牛顿法；边界条件

目录

1.引言

悬链线(Catenary)指的是一种两端固定的一条粗细与质量分布均匀、柔软不能伸长的链条在重力的作用下所具有的曲线形状，例如悬索桥等，因其与两端固定的绳子在均匀引力作用下下垂相似而得名。伽利略曾认为悬链线是一条抛物线，荷兰物理学家惠更斯用物理方法证明了这条曲线不是抛物线，最终约翰·伯努利证明了等高悬链线实际上是一条双曲余弦， $y=\frac{e^x+e^{-x}}{2}$。

我们所研究的悬链线是两端端点分别为点（0,0）和点（1，h）的悬链线。本文试图从系统能量的角度来求解该悬链线的方程，也就是说给定 $y(0)=0, y(1)=h$，求解一条曲线 $y=y(x)$，使得系统的总能量最小，因为稳定状态一定是系统能量最低的状态。本文最终的目的是想要得到该悬链线问题的数值解法，因而可以先考虑非理想条件下的悬链线问题，考虑一个离散的悬链线问题。用离散的悬链线问题来推导悬链线的微分方程。

2.悬链线微分方程的推导

本文选择将铁链作为离散的悬链线问题的研究对象，并非计算一条连续均匀的重绳索自然下垂的情况，而是一条铁链下垂的情况。假设铁链的每一小段之间没有摩擦，每个链条段可以看做一个小段刚体，计算整个铁链的在重力场中的极小势能。

假设一根铁链长度为 L，由 N 个小段刚杆组成，每段刚杆的长度都是Δ， 现在要计算这条铁链在自由悬挂时的形状。根据假设，L = NΔ，其中Δ是每一段刚杆的长度。铁链悬挂在 A 点和 B 点，A 点坐标为 A(0,0),B 点坐标为 B(1,h),0<a<$\sqrt{1+h^2}$. 于是铁链可以用 N+1 个点的坐标来描述。因为铁链共有 N 段，所以铁链共有 N+1 个端点，记作 $p_0=A, p_1,\ p_2, \ldots, p_N$=B. 记一段刚杆$p_{i-1}p_i$, i=1,2,⋯,N 与横轴的夹角为$\theta_i$,于是每一段刚杆可以用矢量表示为:

$$
\begin{aligned}
&\overrightarrow{Op_0} = \overrightarrow{OA} = (0,0) \\
&\overrightarrow{p_0p_1} = \Delta(\cos\theta_1, \sin\theta_1) \\
&\cdots \\
&\overrightarrow{p_{i-1}p_i} = \Delta(\cos\theta_i, \sin\theta_i) \\
&\cdots \\
&\overrightarrow{p_{N-1}p_N} = \Delta(\cos\theta_N, \sin\theta_N) \\
&\overrightarrow{Op_N} = \overrightarrow{O\mathrm{B}} = (\mathrm{a}, 0)
\end{aligned}
$$

由此可以得到第 i 个端点的坐标为

$$
\begin{aligned}
\overrightarrow{p_0p_i} &= \sum_{j=1}^{i} \overrightarrow{p_{j-1}p_j} \\
&= \sum_{j=1}^{i} \Delta(\cos\theta_j, \sin\theta_j) \\
&= \Delta(\sum_{j=1}^{i} \cos\theta_j, \sum_{j=1}^{i} \sin\theta_j), i = 0, 1, 2, \ldots, N
\end{aligned}
$$

因为铁链的两端已经固定在 A(0,0)和 B(1,h)两点，所以有限制条件

$$\sum_{j=1}^{N} cos\theta_i = 1/\Delta$$

$$\sum_{j=1}^{N} sin\theta_i = h/\Delta$$

然后计算每一小段的势能为(y_{i-1}为刚杆左端的纵坐标，y_i为刚杆右端的纵坐标，假设刚杆的质量均匀分布，可以用刚杆中心处的势能代替刚杆的总势能，λ为刚杆的线密度，g 为重力加速度)

$$
\begin{aligned}
E_i &= \lambda\Delta g \frac{1}{2}(y_{i-1} + y_i) \\
&= \frac{1}{2}\lambda\Delta g\Big(\Delta\sum_{j=1}^{i-1}\sin\theta_j + \Delta\sum_{j=1}^{i}\sin\theta_j\Big) \\
&= \lambda\Delta^2 g\Big(\frac{1}{2}\sin\theta_i + \sum_{j=1}^{i-1}\sin\theta_j\Big), i = 1, 2, \ldots, N
\end{aligned}
$$

求和得到铁链的总势能为

$$
\begin{aligned}
E &= \sum_{i=1}^{N} E_i \\
&= \sum_{i=1}^{N} \lambda \Delta^2 g \Big(\frac{1}{2} \sin\theta_i + \sum_{j=1}^{i-1} \sin\theta_j \Big) \\
&= \lambda \Delta^2 g \Big(\frac{1}{2} \sum_{i=1}^{N} \sin\theta_i + \sum_{i=1}^{N} \sum_{j=1}^{i-1} \sin\theta_j \Big)
\end{aligned}
$$

已经知道铁链的两端固定，所以我们有两个约束条件。于是这个能量极小值问题就转化为求解约束情况下的多元函数的极小值，引入拉格朗日乘子ε_1,ε_2，得到

$$
E = \lambda \Delta^2 g \Big(\frac{1}{2} \sum_{i=1}^{N} \sin\theta_i + \sum_{i=1}^{N} \sum_{j=1}^{i-1} \sin\theta_j \Big) + \epsilon_1 \Big(\sum_{i=1}^{N} \cos\theta_i - \frac{a}{\Delta} \Big) + \epsilon_2 \sum_{i=1}^{N} \sin\theta_i
$$

这是一个以θ_i, i=1, 2, ⋯, N 为变量的受约束的多元函数的极值问题。该函数取得极值的必要条件是$\nabla_{\theta_i} E = 0$. 对变量求偏微分，令偏微分等于零，得到

$$
\begin{aligned}
\frac{\partial E}{\partial \theta_k} &= \lambda \Delta^2 g \Big(\frac{1}{2} \cos\theta_k + \sum_{i=k+1}^{N} \cos\theta_k \Big) - \epsilon_1 \sin\theta_k + \epsilon_2 \cos\theta_k \\
&= \Big((1/2 + N - k) \lambda \Delta^2 g + \epsilon_2 \Big) \cos\theta_k - \epsilon_2 \sin\theta_k \\
&= 0, k = 1, 2, \ldots, N
\end{aligned}
$$

于是得到平衡态的角度为

$$
\begin{aligned}
\tan\theta_k &= \frac{(1/2 + N - k) \lambda \Delta^2 g + \epsilon_2}{\epsilon_1} \\
&= (1/2 + N - k)\alpha + \beta, k = 1, 2, \ldots, N
\end{aligned}
$$

其中，

$$
\alpha = \frac{\lambda \Delta^2 g}{\epsilon_1}, \beta = \frac{\epsilon_2}{\epsilon_1}
$$

代入约束条件，得到

$$f(\alpha,\beta)=\sum_{k=1}^{N}\frac{1}{\sqrt{1+((\frac{1}{2}+N-k)\alpha+\beta)^2}}-1/\Delta=0$$

$$g(\alpha,\beta)=\sum_{k=1}^{N}\frac{(\frac{1}{2}+N-k)\alpha+\beta}{\sqrt{1+((\frac{1}{2}+N-k)\alpha+\beta)^2}}-h/\Delta=0$$

要求解这个方程，我们需要借助于牛顿迭代法，也就是

$$\begin{pmatrix} x \\ y \end{pmatrix}=\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}-\begin{pmatrix} \frac{\partial f(x_0,y_0)}{\partial x} & \frac{\partial f(x_0,y_0)}{\partial y} \\ \frac{\partial g(x_0,y_0)}{\partial x} & \frac{\partial g(x_0,y_0)}{\partial y} \end{pmatrix}^{-1}\begin{pmatrix} f(x_0,y_0) \\ g(x_0,y_0) \end{pmatrix}$$

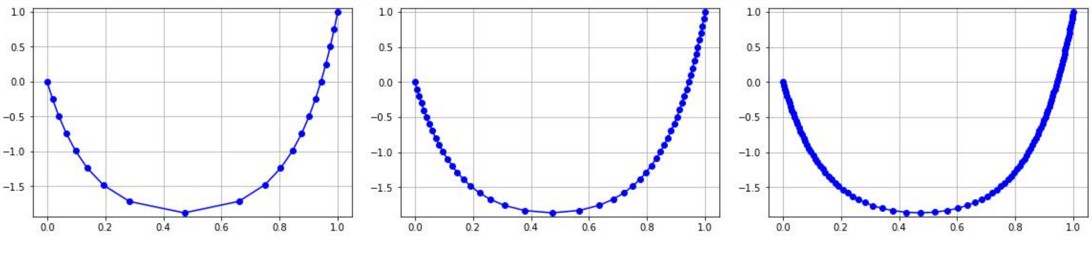
求出α，β就可以得到所有的θ_k=arctan((1/2+N-k)α+β)，进而求出所有的刚杆端点坐标，从而得到铁链的平衡态构型。

3.实验结果及分析

由已知条件，悬链线形状受制于端点坐标(1, h)，悬链线长度 L，另外，在铁链问题中，还受到等分段数 N 影响。

基于控制变量法，有如下结果分析：

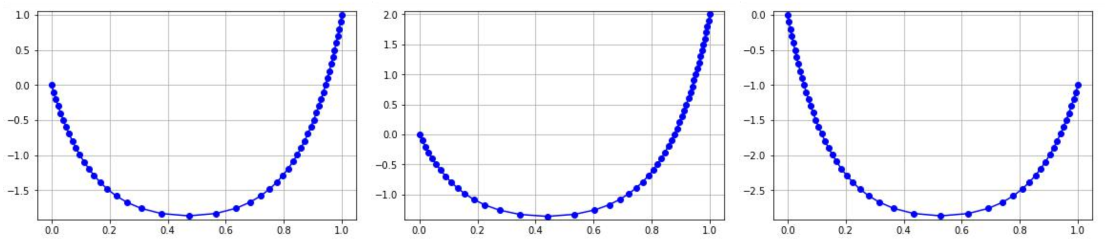
(1) N 的影响(保持 h, L 不变)



从左至右，N = 20, 50, 100

从结果可以看出，N 控制悬链线的“连续性”，换言之，随着 N 的增大，铁链更趋向于绳索。

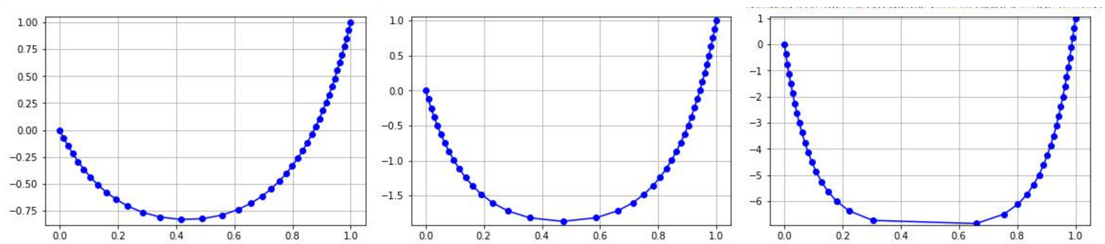
(2) h 的影响(保持 N, L 不变)



从左至右，h = 1, 2, -1

从结果可以看出，h 控制悬链线的形状，控制效果也很直观，就是右端点的高低。

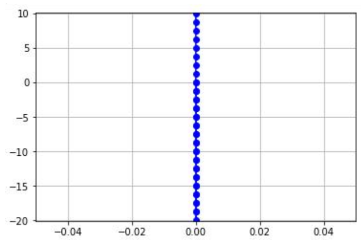
(3) L 的影响



从左至右，L = 3, 5, 15

从结果可以看出，L 对于悬链线的控制较为隐性，似乎是 L 越大，底部越平坦。实际上，L 的控制效果是和 N 紧密联系在一起的，代码中直接起控制效果的是变量 Delta = L / N (即，每段长度)。

当 L 增大时，由于 N 不变，每段的长度也增加，这就导致底部不再是是很多个“小节”组成的曲线，而是只有一节，所以呈现一根略带角度的线段形状。



L = 50

更极端地，当 L 很大的时候，悬链线呈现几乎竖直的形状。

(4) 铁链与绳索对比

我们期待，当铁链有无穷多段的时候，铁链的平衡态构型应该和绳索的是一样的。假设 h = 0，绳索被固定在两端 A(0, 0)，B(a, 0)，绳索的长度为 L，L > a。用变分法很容易算出绳索的解析式为：

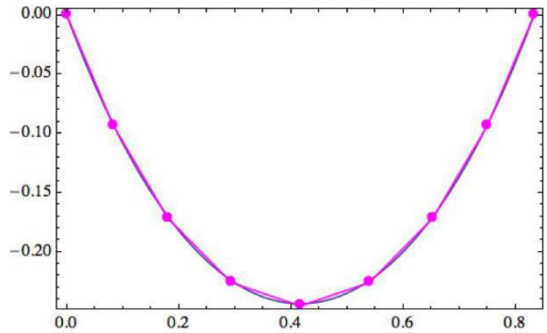
$$y = c\cosh\frac{x-\frac{a}{2}}{c} - c\cosh\frac{a}{2c}$$

因为绳索是不可伸长的，所以有限制条件：

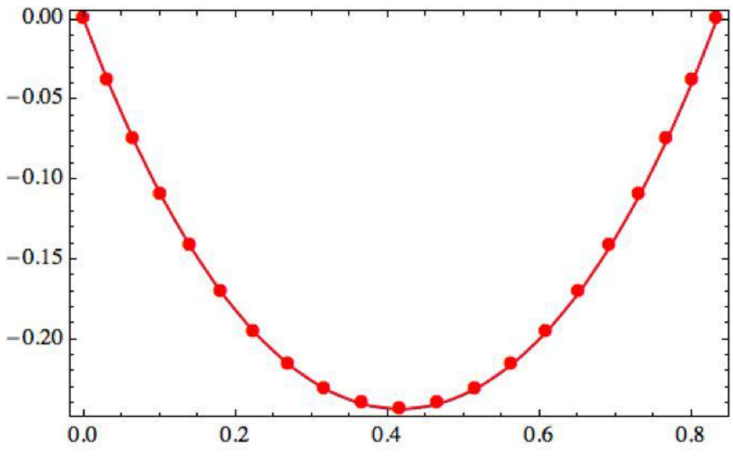
$$2c\sinh\frac{a}{2c} = L$$

假设 L= 1，a = 5/6，可得 c = 0.3913，

当 N = 8 时，绳索与铁链的比较如图：



可以看出，在最底端绳索与铁链稍有不同。进一步增大铁链的节数，得到 N = 20 时，铁链与绳索的比较如图:



可以看到，这时铁链与绳索的平衡态构型已经几乎没有区别了。

4.结语

通过本次 Project，我们对悬链线问题作了深入的研究，并且加深了对牛顿迭代法等数值分析理论知识的理解。我们深深地感受到，Project 是一个非常锻炼人的过程，不同于以往的解题，Project 要求我们在问题分析、资料搜集、理论推导、编程、数据分析、报告撰写方方面面都要下功夫，当完整地完成一个 Project 后，我们小组每个成员都受益匪浅。

感谢单老师的悉心教导，张威助教的辛苦工作，还有我们小组成员的通力合作，我们将在以后的学习生活中，不断运用在《高等数值分析》课上学到的知识。

任务分工

姓名	学号	分工
彭秀枫	11849095	编程与调试； 资料搜集； 报告撰写：摘要、实验结果及分析部分、结语部分。
陈泽彬	11849169	编程与调试； 资料搜集。
尹文壮	11849155	资料搜集； 报告撰写。
陈炫午	11710323	资料搜集； 报告撰写：报告总体、引言部分。
李奕志	11712206	资料搜集； 报告撰写：悬链线微分方程的推导部分。

参考文献

[1] 李恩志.悬链线问题的数值解[EB/OL].https://zhuanlan.zhihu.com/p/29457428

[2] 维基百科.悬链线[EB/OL].https://zh.wikipedia.org/wiki.

[3] ProofWiki.Equation of Catenary[EB/OL].https://proofwiki.org/wiki/Equation_of_Catenary

附录

代码

```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
import math

""" 参数 """
"""
悬链线左端点：(0, 0)
悬链线右端点：(1, h)
悬链线长度: L
等分段数: N
"""
h = 1
L = 5
N = 50

# 其他参数
a = 1 # 悬链线右端点横坐标
Delta = L / N # 每一段长度

def f(alpha, beta):
    s = 0
    for k in range(1, N+1):
        s = s + 1/np.sqrt(1 + ((0.5 + N - k)*alpha + beta)**2)
    s = s - a/Delta
    return s

delta = 1.e-6

def fx(x, y):
    return (f(x + delta, y) - f(x, y))/delta

def fy(x, y):
    return (f(x, y + delta) - f(x, y))/delta

def g(alpha, beta):
    s = 0
    for k in range(1, N+1):
        s = s + ((0.5 + N - k)*alpha + beta)/np.sqrt(1 + ((0.5 + N - k)*alpha + beta)**2)
```

```
        s = s + (h)/Delta
        return s

def gx(x, y):
        return (g(x + delta, y) - g(x, y))/delta

def gy(x, y):
        return (g(x, y+delta) - g(x, y))/delta

def newton(x0, y0):
        count = 0
        iterationMax = 20
        eps = 1.0e-16
        vector = np.zeros(2)
        J = np.zeros((2,2))
        vector[0] = x0
        vector[1] = y0
        fg = np.zeros(2)
        while (count < iterationMax):
                count = count + 1
                old = vector
                x = vector[0]
                y = vector[1]
                J[0,0] = fx(x, y)
                J[0,1] = fy(x, y)
                J[1,0] = gx(x, y)
                J[1,1] = gy(x, y)
                fg[0] = f(x, y)
                fg[1] = g(x, y)
                det = np.linalg.det(J)
                if (abs(det) < 1.e-16):
                        break
                vector = vector - np.linalg.inv(J).dot(fg)
                diff = vector - old
                error = diff.dot(diff)
                #print "count = ", count, ", error = ", error, ", det = ", det
                if (error < eps):
                        break
        return vector

def createCurve(v):
        x = []
        y = []
        theta = []
```

```
    for k in range(1, N+1):
        theta.append(np.arctan((0.5 + N - k)*v[0] + v[1]))
    sx = 0
    sy = 0
    x.append(sx)
    y.append(sy)
    for i in range(len(theta)):
        sx = sx + Delta*np.cos(theta[i])
        sy = sy + Delta*np.sin(theta[i])
        x.append(sx)
        y.append(-(sy))
    return x, y

if __name__=='__main__':
    v = newton(0.6, -5)
    x, y = createCurve(v)
    plt.plot(x, y, "b-o")
#     y_star = x
#     N = len(x)
#     for i in range(N):
#         y_star[i] = math.cosh(x[i] - 1/2) - math.cosh(1/2)
#     plt.plot(x, y_star, "o")
    plt.grid()
    plt.xlim(min(x) - 0.05, max(x) + 0.05)
    plt.ylim(min(y) - 0.05, max(y) + 0.05)
    plt.show()
```