高程控制网平差程序设计

1. 设计背景

水准网中，把3条和3条以上水准路线的交点称为结点，两条水准路线的交点称为节点。水准网既可以按间接平差进行平差，也可以按条件平差进行平差。一般说来，水准网结点之间的节点数量越多，采用条件平差法平差效果较好；反之，结点密集的控制网易采用间接平差法，因此时条件方程太多，不容易一一列出。不过对于复杂的水准网，若使用间接平差方程多，使用条件平差条件方程复杂，有时采用两种方式结合的方法进行，主要有两种方案：一是利用间接平差先求出结点高程，然后按单一水准路线条件平差对各水准路线进行平差；二是先按附有未知数的条件平差对结点进行平差，求得结点未知数后，各线路列独立条件方程，进行平差。

而在测绘软件中，很多问题都要涉及到线性方程组求解和矩阵求逆的问题，常规的克莱姆法则远远满足不了实际的计算需要，如何建立在计算机上可以实现的有效而实用的解法，对测绘软件设计具有极其重要的意义。

1. 开发环境

PyCharm 2019.3.1 (Professional Edition)

Build #PY-193.5662.61, built on December 18, 2019

Licensed to PyCharm Evaluator

Expiration date: January 22, 2020

Runtime version: 11.0.5+10-b520.17 x86\_64

VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

macOS 10.14.6

GC: ParNew, ConcurrentMarkSweep

Memory: 990M

Cores: 4

Language: Python 3.6

1. 程序说明
2. 定义矩阵类，实现矩阵的初始化以及生成对应列表

**class** Matrix:  
 **def** \_\_init\_\_(self):  
 self.Row = 0  
 self.Column = 0  
 self.M = []  
  
 **def** blank\_in(self):  
 M0 = []  
 **for** i **in** range(1, self.Row + 1):  
 x = input(**"请输入第{}行数据："**.format(i))  
 x0 = list(map(float, x.rstrip().split()))  
 M0.append(x0)

print(**""**)  
 self.M = M0

定义的矩阵类将初始化一个带有Row和Column两个整型参数，一个列表M的对象，并使用blank\_in完成对矩阵的输入，并将输入的数据存放到一个Column \* Row的列表中。

1. 矩阵求逆算法实现

**def** Inverse(self):  
 **def** Identity(m):  
 n = len(m)  
 l = []  
 **for** i **in** range(n):  
 l.append([])  
 **for** j **in** range(n):  
 **if** i == j:  
 l[i].append(1)  
 **else**:  
 l[i].append(0)  
 **return** l  
  
 **def** Swap(m):  
 n = len(m)  
 swap = []  
 l = []  
 **for** i **in** range(n):  
 swap.append(i)  
 l.append([])  
 **for** j **in** range(n):  
 l[i].append(0)  
  
 **for** i **in** range(n):  
 max\_row = m[i][i]  
 row = i  
 **for** j **in** range(i, n):  
 **if** m[j][i] >= max\_row:  
 max\_row = m[j][i]  
 row = j  
 swap[i] = row  
  
 **if** row != i:  
 **for** j **in** range(n):  
 m[i][j], m[row][j] = m[row][j], m[i][j]  
  
 **for** j **in** range(i + 1, n):  
 **if** m[j][i] != 0:  
 l[j][i] = m[j][i] / m[i][i]  
 **for** k **in** range(0, n):  
 m[j][k] = m[j][k] - (l[j][i] \* m[i][k])  
  
 **return** swap, m, l  
  
 **def** Step(m):  
 n = len(m)  
 long = len(m) - 1  
 l = []  
 **for** i **in** range(n):  
 l.append([])  
 **for** j **in** range(n):  
 l[i].append(0)  
  
 **for** i **in** range(n-1):  
 **for** j **in** range(long-i):  
 **if** m[long-i-j-1][long-i] != 0 **and** m[long-i][long-i] != 0:  
 l[long-i-j-1][long-i] = m[long-i-j-1][long-i] / m[long-i][long-i]  
 **for** k **in** range(n):  
 m[long-i-j-1][k] = m[long-i-j-1][k] - l[long-i-j-1][long-i] \* m[long-i][k]  
  
 **return** m, l  
  
 **def** Diagonal(m):  
 n = len(m)  
 l = []  
 **for** i **in** range(n):  
 l.append(m[i][i])  
 **return** l  
  
 **def** Gauss\_Jordan(matrix):  
 n = len(matrix)  
 new = Identity(matrix)  
 (swap, matrix1, l1) = Swap(matrix)  
 (matrix2, l2) = Step(matrix1)  
 l3 = Diagonal(matrix2)  
 **for** i **in** range(n):  
 **if** swap[i] != i:  
 new[i], new[swap[i]] = new[swap[i]], new[i]  
 **for** j **in** range(i+1, n):  
 **for** k **in** range(n):  
 **if** l1[j][i] != 0:  
 new[j][k] = new[j][k] - l1[j][i] \* new[i][k]  
 **for** i **in** range(n-1):  
 **for** j **in** range(n-i-1):  
 **if** l2[n-1-i-j-1][n-1-i] != 0:  
 **for** k **in** range(n):  
 new[n-1-i-j-1][k] = new[n-1-i-j-1][k]-l2[n-1-i-j-1][n-i-1] \* new[n-1-i][k]  
 **try**:  
 **for** i **in** range(n):  
 **for** j **in** range(n):  
 new[i][j] = new[i][j] / l3[i]  
 **return** new  
 **except** ZeroDivisionError:  
 print(**"该矩阵不可逆"**)  
  
 **return** Gauss\_Jordan(t.M)

高斯消去法有很多变形，选主元法即是其中之一，在选主元素法的基础上还可以对算法进行修改：在消元过程中选主元后，先将主元化为1，而后将主元所在列上、下方各元素均化为0，这样消元的结果使系数矩阵化成了单位阵，无需回代就得到了原方程的解。这种无回代过程的选主元素法称为高斯-约当（Gauss-Jordan）消去法。高斯-约当法求逆矩阵十分方便，因为消元过程实质上就是对增广矩阵实行初等变换。该方法放置于矩阵类内，方便其随时调用。

1. 具体平差程序

f = open(**r'/Users/apple/Documents/test/con.txt'**)  
lines = f.readlines()  
f.close()  
  
w = int(lines[1])  
H = list(map(float, lines[3].rstrip().split()))  
g = list(map(float, lines[4].rstrip().split()))  
j = list(map(float, lines[5].rstrip().split()))  
  
A = Matrix()  
A.Row = len(g) - w  
A.Column = len(g)  
A.blank\_in()  
AA = np.mat(A.M)  
  
W = Matrix()  
W.Row = len(g) - w  
W.Column = 1  
W.blank\_in()  
WW = np.mat(W.M)  
  
P = []  
**for** i **in** range(len(j)):  
 P.append([])  
 **for** ii **in** range(len(j)):  
 P[i].append(0)  
**for** i **in** range(len(j)):  
 P[i][i] = j[i]  
PP = np.mat(P)  
  
Naa = AA \* PP \* AA.T  
KK = -Naa.I \* WW  
VV = PP \* AA.T \* KK  
  
V = VV.tolist()  
**for** i **in** range(len(g)):  
 g[i] += (V[i][0] / 1000)  
 print(**"h{}的改正后的结果为{}"**.format(i + 1, g[i]))  
  
VTPV = KK.T \* Naa \* KK  
sigma = math.sqrt(VTPV[0, 0] / (len(g) - w))  
print(**"精度评定："**)  
print(**"单位权中误差为：{}"**.format(sigma))

该部分是平差的具体实例，通过调用Matrix对象内的列表，对列表转为numpy中的矩阵类型来实现对矩阵的运算操作。前期数据来源为con.txt，其内容如下：

导线 正常高 距离权重

2

HA HB h1 S1 h2 S2 h3 S3 h4 S4 HC HD

12.013 10.013

-1.004 1.516 2.512 1.520

2 1 2 1.5

数据与数据之间，点号与点号之前用空格隔开，方便后期对数据进行分割及处理，内容标准按照指导书约定编写。

1. 使用说明

运行程序后会让输入矩阵A及W的相关信息：



完成输入后即可直接导出改正后的数值以及单位权中误差：



可在不同情况下修改con.txt的内容来完成对不同数据及情况的条件平差，实验中所使用的数据来自：

