以下为自定义模块

# findLocal.py

# 找出音爆时的位置与时间

from scipy.optimize import minimize

import numpy as np

import alltime

def find\_local(X, Y, Z, method="SLSQP", file=None, L=None, constraints : bool = False, funMethod : str = 'square', x0 : list = [42, -14, 1.4, 5.72], ifprint : bool = True):

    with open('localData.csv', 'w', encoding='gb2312') as fp:

        if file != None:

            dataNumber = len(alltime.all\_time(file))

            print(dataNumber)

        else:

            dataNumber = 1

        for ilenth in range(dataNumber):

            if file != None:

                L = alltime.all\_time(file)[ilenth]

            if funMethod == 'abs':

                fun = lambda x: sum(abs((((X[i] - x[0])\*\*2 + (Y[i] - x[1])\*\*2 + (Z[i] - x[2])\*\*2)\*\*0.5 - (L[i] + x[3]))) for i in range(1, len(X)))

            elif funMethod == 'square':

                fun = lambda x: sum((((X[i] - x[0])\*\*2 + (Y[i] - x[1])\*\*2 + (Z[i] - x[2])\*\*2)\*\*0.5 - (L[i] + x[3]))\*\*2 for i in range(1, len(X)))

            cons = ({'type': 'ineq', 'fun': lambda x: x[0] + 100},

                    {'type': 'ineq', 'fun': lambda x: -x[0] + 100},

                    {'type': 'ineq', 'fun': lambda x: x[1] + 100},

                    {'type': 'ineq', 'fun': lambda x: -x[1] + 100},

                    {'type': 'ineq', 'fun': lambda x: x[2] + 0},

                    {'type': 'ineq', 'fun': lambda x: -x[2] + 10},

                    {'type': 'ineq', 'fun': lambda x: x[3] + 100},

                    {'type': 'ineq', 'fun': lambda x: -x[3] + 500}

                )

            if constraints == False:

                cons = ()

            x0 = np.array(x0)

            if method == 'SLSQP':

                res = minimize(fun, x0, method='SLSQP', constraints=cons)

            elif method == "BFGS":

                res = minimize(fun, x0, method='BFGS')

            if ifprint == True:

                print("最小值:", res.fun)

                print("最优解:", res.x)

                print('经度' + str(res.x[0] / 97.304 + 110.241))

                print('纬度' + str(res.x[1] / 111.263 + 27.204))

                print('高度' + str(res.x[2]))

                print('时间' + str(res.x[3] / 0.34))

            if res.success:

                fp.write(str(ilenth) + ',最小值,' + str(res.fun) + ',最优解,' + str(res.x))

                fp.write('\n')

# findAllCombination.py

# 找出所有时间组合方式

from itertools import product

import numpy as np

def find\_all(B, C, D, E, F, G):

    name = str(len(B) + 1) + '.csv'

    nametime = str(len(B) + 1) + 'time.csv'

    A = [100.767, 164.229, 214.85, 270.065]

    all\_combinations = list(product(B, C, D, E, F, G))

    with open(name, 'w', encoding='utf-8') as fp:

        for combination in all\_combinations:

            templist = list(combination)

            templist.insert(0, A[-len(B)])

            np\_list = np.array(templist)

            templist = (np\_list\*0.34).tolist()

            templist.insert(0, None)

            fp.write(str(templist))

            fp.write('\n')

    with open(nametime, 'w', encoding='utf-8') as fp:

        for combination in all\_combinations:

            templist = list(combination)

            templist.insert(0, A[-len(B)])

            np\_list = np.array(templist)

            templist = (np\_list).tolist()

            templist.insert(0, None)

            fp.write(str(templist))

            fp.write('\n')

    return name

# drawpic.py

# 画出监测塔与音爆点三维图

import numpy as np

import matplotlib.pyplot as plt

def drawpic(X, Y, Z, point : tuple, ifname=False, Ltime=np.array([]), towernum=None, L=None):

    if Ltime != np.array([]):

        L = (Ltime\*0.34 - 4.07997).tolist()

        L.insert(0, None)

        print(L)

    color = ['b', 'g', 'r', '#39CC4B', 'c', 'm', 'y']

    centerlist = [(X[i+1], Y[i+1], Z[i+1]) for i in range(7)]

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    for i in range(7):

        center = centerlist[i]

        radius = L[i+1]

        print(center, radius)

        ax.scatter(center[0], center[1], center[2], color=color[i], s=100, zorder=100000, alpha=0.3)

        if ifname:

            ax.scatter(center[0], center[1], center[2], color=color[i], s=100, zorder=100000, alpha=0.3, label='tower' + str(i+1))

        theta = np.linspace(0, 2\*np.pi, 100)

        phi = np.linspace(0, np.pi, 50)

        theta, phi = np.meshgrid(theta, phi)

        x = center[0] + radius \* np.sin(phi) \* np.cos(theta)

        y = center[1] + radius \* np.sin(phi) \* np.sin(theta)

        z = center[2] + radius \* np.cos(phi)

        ax.plot\_wireframe(x, y, z, color=color[i], alpha=0.1, zorder=2)

    ax.scatter(point[0], point[1], point[2], color='black', s=500, marker='\*', zorder=100000)

    if ifname:

        ax.scatter(point[0], point[1], point[2], color='black', s=100, marker='\*', zorder=100000, label= 'boom' + str(towernum))

        plt.legend(loc='upper left')

    ax.set\_xlabel('X')

    ax.set\_ylabel('Y')

    ax.set\_zlabel('Z')

    plt.show()

# alltime.py

# 读取时间组合文件

def all\_time(file):

    with open(file, 'r') as file:

        lines = file.readlines()

    list\_of\_lists = []

    for line in lines:

        list\_from\_line = eval(line.strip())

        list\_of\_lists.append(list\_from\_line)

    return list\_of\_lists

# afterFindAandC.py

# 找出所有时间组合的误差后选出误差最小的组合

import csv

def findAandC(filename):

    with open(filename, 'r', encoding='gb2312') as file:

        csv\_reader = csv.reader(file)

        min\_value = float('inf')

        min\_row = None

        for row in csv\_reader:

            value = float(row[2])

            if value < min\_value:

                min\_value = value

                min\_row = row

    final = min\_row[-1]

    final = final.replace("[", "")

    final = final.replace("]", "")

    data = [x for x in final.split(' ') if x != '']

    print(data)

    if min\_row:

        print("最小值:", min\_value)

        print("最优解:", min\_row[-1])

        print('经度' + str(float(data[0]) / 97.304 + 110.241))

        print('纬度' + str(float(data[1]) / 111.263 + 27.204))

        print('高度' + str(float(data[2])))

        print('时间' + str(float(data[3]) / 0.34))

    else:

        print("未找到最小值")

    return min\_row[0]

# q4findLocal.py

# 修正后找出音爆时的位置与时间

from scipy.optimize import minimize

import numpy as np

import alltime

def find\_local(X, Y, Z, method="SLSQP", file=None, L=None, constraints : bool = False, funMethod : str = 'square', x0 : list = [42, -14, 1.4, 5.72], ifprint : bool = True, filename = 'localData.csv'):

    with open(filename, 'a+', encoding='utf-8') as fp:

        if file != None:

            dataNumber = len(alltime.all\_time(file))

            print(dataNumber)

        else:

            dataNumber = 1

        for ilenth in range(dataNumber):

            if file != None:

                L = alltime.all\_time(file)[ilenth]

            if funMethod == 'abs':

                fun = lambda x: sum(abs((((X[i] - x[0])\*\*2 + (Y[i] - x[1])\*\*2 + (Z[i] - x[2])\*\*2)\*\*0.5 - (L[i] + x[3]))) for i in range(1, len(X)))

            elif funMethod == 'square':

                fun = lambda x: sum((((X[i] - x[0])\*\*2 + (Y[i] - x[1])\*\*2 + (Z[i] - x[2])\*\*2)\*\*0.5 - (L[i] + x[3]))\*\*2 for i in range(1, len(X)))

            cons = ({'type': 'ineq', 'fun': lambda x: x[0] + 100},

                    {'type': 'ineq', 'fun': lambda x: -x[0] + 100},

                    {'type': 'ineq', 'fun': lambda x: x[1] + 100},

                    {'type': 'ineq', 'fun': lambda x: -x[1] + 100},

                    {'type': 'ineq', 'fun': lambda x: x[2] + 0},

                    {'type': 'ineq', 'fun': lambda x: -x[2] + 20},

                    {'type': 'ineq', 'fun': lambda x: x[3] + 100},

                    {'type': 'ineq', 'fun': lambda x: -x[3] + 500}

                )

            if constraints == False:

                cons = ()

            x0 = np.array(x0)

            if method == 'SLSQP':

                res = minimize(fun, x0, method='SLSQP', constraints=cons)

            elif method == "BFGS":

                res = minimize(fun, x0, method='BFGS')

            if ifprint == True:

                print("最小值:", res.fun)

                print("最优解:", res.x)

                print('经度' + str(res.x[0] / 97.304 + 110.241))

                print('纬度' + str(res.x[1] / 111.263 + 27.204))

                print('高度' + str(res.x[2]))

                print('时间' + str(res.x[3] / 0.34))

            if res.success:

                fp.write(str((res.x).tolist())[1:-2])

                fp.write('\n')

# q4plot.py

# 添加随机误差后坐标与时间分布

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

def pltN(file, colname, boomname, step):

    temp = 0

    name = ['x', 'y', 'z', 't']

    col = name.index(colname)

    for i in [col]:

        df = pd.read\_csv(file)

        if name[col] == 't':

            y\_values = df.iloc[:, i] / (-0.34)

        else:

            y\_values = df.iloc[:, i]

        y\_values\_sorted = y\_values.sort\_values()

        min\_value = y\_values\_sorted.min()

        interval\_counts = {}

        for value in y\_values\_sorted:

            interval = int((value - min\_value) / step)

            if interval not in interval\_counts:

                interval\_counts[interval] = 1

            else:

                interval\_counts[interval] += 1

        interval\_starts = []

        interval\_ends = []

        counts = []

        for interval, count in interval\_counts.items():

            interval\_start = min\_value + interval \* step

            interval\_end = interval\_start + step

            interval\_starts.append(interval\_start)

            interval\_ends.append(interval\_end)

            counts.append(count)

        plt.bar(interval\_starts, counts, width=step, align='edge', edgecolor='red')

        plt.xlabel(name[col])

        plt.title(boomname)

        plt.show()

        temp = temp + 1

以下为各问题代码

# q1.py

# 第一问找出音爆点位置

import findLocal

X = [None, 0, 52.44686, 45.83018, 0.97304, 27.53703, 21.9907, -18.887]

Y = [None, 0, 28.03828, 64.6438, 69.09432, 45.95162, 79.77557, -9.23483]

Z = [None, 0.824, 0.727, 0.742, 0.85, 0.786,0.678, 0.575]

L = [None, 34.26078, 38.1548, 63.9268, 88.0549, 40.27062, 90.73614, 55.42816]

print('七个监测站')

findLocal.find\_local(X, Y, Z, L=L, method='SLSQP', funMethod='abs', constraints=True, x0=[41, -14, 5.5, 1])

for i in [X, Y, Z, L]:

    i.pop(-3)

print('\n去除误差较大的检测塔E')

findLocal.find\_local(X, Y, Z, L=L, method='SLSQP', funMethod='abs', constraints=True)

# q1plot.py

# 画出监测塔与音爆点三维图

import drawpic

# 以A检测塔为原点

X = [None, 0, 52.44686, 45.83018, 0.97304, 27.53703, 21.9907, -18.887]

Y = [None, 0, 28.03828, 64.6438, 69.09432, 45.95162, 79.77557, -9.23483]

Z = [None, 0.824, 0.727, 0.742, 0.85, 0.786, 0.678, 0.575]

L = [None, 34.26078, 38.1548, 63.9268, 88.0549, 40.27062, 90.73614, 55.42816]

drawpic.drawpic(X, Y, Z, L=L, point=(42.0187, -14.5705,   1.1324))

# q3.py

# 匹配四组时间并求出四个音爆发生的位置与时间

import findAllCombination

import findLocal

import afterFindAandC

import csv

from itertools import islice

B = [92.453, 112.22, 169.362, 196.583]

C = [75.560, 110.696, 156.936, 188.02]

D = [94.653, 141.409, 196.517,258.985]

E = [78.600, 86.216, 118.443, 126.669]

F = [67.274, 166.270, 175.482, 266.871]

G = [103.738, 163.024, 206.789, 210.306]

X = [None, 0, 52.73877, 50.69538, 0.97304, 27.53703, 21.9907, -18.87698]

Y = [None, 0, 28.03828, 64.6438, 91.34692, 45.95162, 97.57765, 35.27037]

Z = [None, 0.824, 0.727, 0.742, 0.85, 0.786, 0.678, 0.575]

for i in range(3):

    name = findAllCombination.find\_all(B, C, D, E, F, G)

    findLocal.find\_local(X, Y, Z, file=name, x0=[0, 0, 13, 0], ifprint=False)

    line = afterFindAandC.findAandC('localData.csv')

    with open(str(len(B) + 1) + 'time.csv', 'r', encoding='utf-8') as file:

        mycsv = csv.reader(file)

        for i, row in enumerate(mycsv):

            if i == int(line):

                tagrow = row

                break

        tagrow = [(s.strip("[] '")) for s in tagrow if s.strip("[] '")]

        tagrow.pop(0)

        tagrow = [float(num) for num in tagrow]

        tagrow.pop(0)

        file.close()

    for i in range(len([B, C, D, E, F, G])):

        [B, C, D, E, F, G][i].remove(tagrow[i])

# q3plot.py

# 依次画出监测塔与四次音爆的三维图

import drawpic

import numpy as np

# 以A检测塔为原点

X = [None, 0, 52.73877, 50.69538, 0.97304, 27.53703, 21.9907, -18.87698]

Y = [None, 0, 28.03828, 64.6438, 91.34692, 45.95162, 97.57765, 35.27037]

Z = [None, 0.824, 0.727, 0.742, 0.85, 0.786, 0.678, 0.575]

# 求得的各音爆时间组合

L1time = np.array([100.767, 112.22, 188.02, 258.985, 118.443, 266.871, 163.024])

L2time = np.array([164.229, 169.362, 156.936, 141.409, 86.216, 166.27, 103.738])

L3time = np.array([214.85, 92.453, 75.56, 196.517, 78.6, 175.482, 210.306])

L4time = np.array([270.065, 196.583, 110.696, 94.653, 126.669, 67.274, 206.789])

# 求得的各音爆相对检测塔A的坐标

point1 = (25.201879, 11.7936807, 12.513927)

point2 = (5.741, 49.6232, 11.4796)

point3 = (44.662257, 49.62316,13.46793)

point4 = (25.20158745, 83.00192217, 11.52885)

drawpic.drawpic(X, Y, Z, point=point1, Ltime=L1time, ifname=True, towernum='1')

drawpic.drawpic(X, Y, Z, point=point2, Ltime=L2time, ifname=True, towernum='2')

drawpic.drawpic(X, Y, Z, point=point3, Ltime=L3time, ifname=True, towernum='3')

drawpic.drawpic(X, Y, Z, point=point4, Ltime=L4time, ifname=True, towernum='4')

# q4.py

# 添加随机误差修正后定位残骸与绘图

import numpy as np

import q4findLocal

import q4plot

boom1 = np.array([100.767, 112.22, 188.02, 258.985, 118.443, 266.871, 163.024])

boom2 = np.array([164.229, 169.362, 156.936, 141.409, 86.216, 166.27, 103.738])

boom3 = np.array([214.85, 92.453, 75.56, 196.517, 78.6, 175.482, 210.306])

boom4 = np.array([270.065, 196.583, 110.696, 94.653, 126.669, 67.274, 206.789])

X = [None, 0, 52.73877, 50.69538, 0.97304, 27.53703, 21.9907, -18.87698]

Y = [None, 0, 28.03828, 64.6438, 91.34692, 45.95162, 97.57765, 35.27037]

Z = [None, 0.824, 0.727, 0.742, 0.85, 0.786, 0.678, 0.575]

mean = 0

std\_dev = 0.5

def boomAddError(boomName, csvname):

    for i in range(1000):

        error = np.random.normal(mean, std\_dev, size=len(boomName))

        boom\_with\_error = boom1 + error

        boom\_with\_error = (boom\_with\_error\*0.34).tolist()

        boom\_with\_error.insert(0, None)

        filename = csvname + '.csv'

        q4findLocal.find\_local(X, Y, Z, L=boom\_with\_error, filename=filename, ifprint=False)

    return filename

q4plot.pltN(boomAddError(boom1, 'boom1'), 'x', 'boom1', 0.05)