代表性自定义模块代码

# 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')

# 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]

# 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