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
```