```
#!/usr/bin/python
#coding=utf-8
"""
Created on Fri Mar 31 20:26:12 2017

@author: Xianyun Zhang
"""

import os
import math
import numpy as np

print
print '#-----------------------Input data----------------------#'
print

start_Time = float(raw_input("Please input start time : "))
end_Time = float(raw_input("Please input end time : "))
wavePeriod = float(raw_input("Please input wave period : "))
waveNumber = float(raw_input("Please input the wave number : "))
deltaL = float(raw_input("Please input the the space of two probe : "))
fft_Num = int(raw_input("Please input FFT number : "))

pathname = os.path.abspath('.')
print pathname
Out_File = open(os.path.join(pathname,outFile.txt), 'w')  #存储屏幕输出信息
Out_Eta = open(os.path.join(pathname,outEta.txt, 'w')  #存储处理后的两个探针数据
DataPath = os.path.join(pathname,'Data')  #探针数据所在目录此处目录提示错误

a = os.listdir(DataPath) #获得探针名称的列表
a.sort() #探针名称排序

fileR = open(os.path.join(DataPath,a[0]), 'r') #打开第一个探针文件
Data = fileR.read() #读取数据
fileR.close()
Data = Data.split('\n') #分行
Data_row=len(Data)-1      #获得行数

raw_Data=np.zeros((Data_row,8))  #创建 Data_row 行 8 列 的数组
for i in range(len(a)):                  #读取探针数据到数组
    filePath = os.path.join(DataPath,a[i])
    raw_Data[...,2*i:2*i+2] = np.loadtxt(filePath)
```

```
raw_Data=np.delete(raw_Data,range(2,8,2),1) #删除 3 5 7 列重复的时间数据

for i in range(Data_row-1):   #获得起始、结束时间的下表
    T1=raw_Data[i,0]
    T2=raw_Data[i+1,0]
    if(T1<=start_Time and T2>start_Time):
        numStart=i
        print 'numStart',i
        Out_File.writelines('numStart '+ str(i) +'\n')
    if(T1<end_Time and T2>=end_Time):
        numEnd=i+1
        print 'numEnd',i+1
        Out_File.writelines('numEnd '+ str(i+1) +'\n')
raw_Data=np.delete(raw_Data,range(numEnd+1,Data_row),0)   #删除结束时间后的数据
raw_Data=np.delete(raw_Data,range(0,numStart),0)   #删除开始时间前的数据

print
print '#----------------Solve average wave period----------------#'
print

Period=np.zeros((4))   #创建包含四个数据（0.0）的数组

mm=np.zeros((500,4)) #创建包含 500 行、4 列的数组
mk=np.zeros([4],dtype = int) #创建包含四个数据（0）的数组 int 类型

for i in   range(1,5):   #数组的 1 2 3 4 四个探针数据循环
    k1=0;Time=np.zeros((0))
    for j in range(raw_Data.shape[0]-1):   #所有的行循环
        T1=raw_Data[j,i]
        T2=raw_Data[j+1,i]
        if(T1<=0 and T2>=0):
            Time=np.append(Time,[raw_Data[j,0]])
            mm[mk[i-1],i-1]=j   #存储上跨的下标
            mk[i-1]=mk[i-1]+1   #存储上跨个数（波浪个数-1）
            k1=k1+1
        #print 'k1= ',k1
    k2=0;T3=0.0
    for k2 in range(k1-1):
        t1=Time[k2]
        t2=Time[k2+1]
        T3=T3+(t2-t1)
    Period[i-1]=T3/(k1-1)
    print 'The period of Num.',i,' probe is : ',Period[i-1]
    Out_File.writelines('The period of Num. '+ str(i) +' probe is : '+str(Period[i-1])+'\n')
```

```
print
print '#----------------Solve average wave height----------------#'
print

aveH=np.zeros((4))

mm=np.delete(mm,range(np.min(mk),500),0) #删除数组中 四个探针中最少的波浪个数 至 500 行的数据
maxH=np.zeros((np.min(mk)-1))  #存储波峰
minH=np.zeros((np.min(mk)-1))  #存储波谷

for i in  range(1,5):
    for j in range(mm.shape[0]-1):
        maxH[j]=np.max(raw_Data[mm[j][i-1]:mm[j+1][i-1],i])#得到临近的两个上跨零点的最大值 即波峰
        minH[j]=np.min(raw_Data[mm[j][i-1]:mm[j+1][i-1],i])#得到临近的两个上跨零点的最小值 即波谷
        aveH[i-1]=(np.sum(maxH)-np.sum(minH))/(np.min(mk)-1)#平均波高

    print 'The average wave height of Num.',i,' probe is : ',aveH[i-1]
    Out_File.writelines('The average wave height of Num. '+ str(i) +' probe is : '+str(aveH[i-1])+'\n')

print
print '#----------------------Interpolation----------------------#'
print

num1=int((end_Time-start_Time)/wavePeriod)-1 #时间范围内的整数个周期

if (start_Time+num1*wavePeriod) > end_Time : #判断结束时间是否超出范围
  print 'Fetal error in the time interval ! '

raw_Time=raw_Data[:,0]#存储时间数据
raw_eta1=raw_Data[:,1]#存储 #1 探针 波面数据
raw_eta2=raw_Data[:,2]#存储 #2 探针 波面数据

fft_Time=np.zeros((fft_Num))#存储处理后的 fft_Num 个时间数据
fft_eta1=np.zeros((fft_Num))#存储处理后的 fft_Num 个 #1 探针 波面数据
fft_eta2=np.zeros((fft_Num))#存储处理后的 fft_Num 个 #2 探针 波面数据

fft_Dt=num1*wavePeriod/(fft_Num-1)#时间步长

fft_Time[0]=start_Time
fft_eta1[0]=raw_Data[0,1]
```

```
fft_eta2[0]=raw_Data[0,2]

for i in range(1,fft_Num):    #得到 fft_Num 个时间序列
    fft_Time[i]=fft_Time[i-1]+fft_Dt

for i in range(1,fft_Num):#得到 fft_Num 个波面序列

    for j in range(raw_Time.shape[0]-1):

        if (raw_Time[j]<=fft_Time[i] and fft_Time[i]<raw_Time[j+1]):
            fft_k=(fft_Time[i]-raw_Time[j])/(raw_Time[j+1]-raw_Time[j])
            fft_eta1[i]=fft_k*(raw_eta1[j+1]-raw_eta1[j])+raw_eta1[j]
            fft_eta2[i]=fft_k*(raw_eta2[j+1]-raw_eta2[j])+raw_eta2[j]
            Out_Eta.writelines(str(fft_eta1[i])+' '+str(fft_eta2[i])+'\n')


print
print '#-------------Solve the reflection coefficient-------------#'
print

#根据 Goda 两点法求解入射波高和反射波高

waveOmega=2*math.pi/wavePeriod

A1=0.;   A2=0.;   B1=0.;   B2=0.;

for i in range(fft_Num-1):    #得到傅里叶系数 A1 A2 B1 B2
    A1=A1+(                    fft_eta1[i]*math.cos(waveOmega*(i)*fft_Dt)                    +
fft_eta1[i+1]*math.cos(waveOmega*(i+1)*fft_Dt) )/2
    B1=B1+(                    fft_eta1[i]*math.sin(waveOmega*(i)*fft_Dt)                    +
fft_eta1[i+1]*math.sin(waveOmega*(i+1)*fft_Dt) )/2
    A2=A2+(                    fft_eta2[i]*math.cos(waveOmega*(i)*fft_Dt)                    +
fft_eta2[i+1]*math.cos(waveOmega*(i+1)*fft_Dt) )/2
    B2=B2+(                    fft_eta2[i]*math.sin(waveOmega*(i)*fft_Dt)                    +
fft_eta2[i+1]*math.sin(waveOmega*(i+1)*fft_Dt) )/2

A1=2*A1/(fft_Num-1);        A2=2*A2/(fft_Num-1);
B1=2*B1/(fft_Num-1);        B2=2*B2/(fft_Num-1);

#得到入射波幅和反射波幅
ampIn=math.sqrt((A2-A1*math.cos(waveNumber*deltaL)-
B1*math.sin(waveNumber*deltaL))**2.0+(B2+A1*math.sin(waveNumber*deltaL)-
B1*math.cos(waveNumber*deltaL))**2)/(2.0*abs(math.sin(waveNumber*deltaL)))
ampRe=math.sqrt((A2-
```

```
A1*math.cos(waveNumber*deltaL)+B1*math.sin(waveNumber*deltaL))**2.0+(B2-
A1*math.sin(waveNumber*deltaL)-
B1*math.cos(waveNumber*deltaL))**2)/(2.0*abs(math.sin(waveNumber*deltaL)))
Refco=ampRe/ampIn    #反射系数

print 'Incident wave hight is : ',2*ampIn
print 'Reflection wave hight is : ',2*ampRe
print 'Reflection coefficient is : ',Refco

Out_File.writelines('Incident wave hight is : '+ str(2*ampIn) +'\n')
Out_File.writelines('Reflected wave hight is : '+ str(2*ampRe) +'\n')
Out_File.writelines('Reflection coefficient is : '+ str(Refco) +'\n')

print
print '#------------Solve the transmission coefficient------------#'
print

print 'Transmission coefficient is : ',aveH[2]/(2*ampIn),' (probe3)'
print 'Transmission coefficient is : ',aveH[3]/(2*ampIn),' (probe4)'

Out_File.writelines('Transmission coefficient is : '+ str(aveH[2]/(2*ampIn)) +' (probe3)'+'\n')
Out_File.writelines('Transmission coefficient is : '+ str(aveH[3]/(2*ampIn)) +' (probe4)' +'\n')

Out_File.close()
Out_Eta.close()
print
print '#-------------------------Finished-------------------------#'
print

raw_input("Press <Enter>...")
```