**Comput.py**

import numpy as np

import pandas as pd

import xlrd

from pprint import pprint

from xlwt import \*

file = Workbook(encoding='utf-8')

table = file.add\_sheet('sheet1')

data = pd.read\_excel('./Resource/data2014.xlsx')

#data = pd.read\_excel('./Resource/exceptEnvironment.xlsx')

#print(data.values.shape)

nrows, ncols = data.values.shape

#print(nrows, ncols)

array = np.array(data.values)

X0 = np.ones(4)

dij = np.array(data.values)

for row in range(nrows - 3):

for col in range(1, ncols):

array[row][col] = (array[row][col] - array[nrows-3][col]) / array[nrows-1][col]

dij[row][col] = abs(array[row][col] - 1)

#print(array[:-3, :][0])

#print(len(array))

#print(dij)

tmp = dij[:-3,1:]

#print(tmp)

a = tmp.min()

b = tmp.max()

p = 0.5

#print('a',a)

#print('b',b)

xg\_y = np.array(data.values)

for row in range(nrows - 3):

for col in range(1, ncols):

xg\_y[row][col] = (a + b\*p) / (dij[row][col] + b\*p)

#print(xg\_y)

xg\_Z = np.zeros(ncols)

for col in range(1, ncols):

for row in range(nrows - 3):

xg\_Z[col] = xg\_Z[col] + xg\_y[row][col]

xg\_Z[col] = xg\_Z[col] / (nrows-3)

xg\_Z = xg\_Z[1:]

#print(xg\_Z)

xg\_Z\_sum = np.sum(xg\_Z)

W = np.zeros(ncols)

for index in range(len(xg\_Z)):

W[index] = xg\_Z[index] / xg\_Z\_sum

#print(W)

G = np.array(data.values[:nrows-3,:2])

for row in range(nrows - 3):

G[row][1] = 0

for col in range(1, ncols):

G[row][1] = G[row][1] + (array[row][col]\*W[col-1])

G[row][1] = 1 - G[row][1]

print(G)

#for index in range(len(G)):

# table.write(index, 0, G[index,0])

# table.write(index, 1, G[index,1])

#file.save('Q:\\ariescc\\mcm\\out.xls')

#fragile = list()

#for index in range(len(G)):

# dic = dict()

# dic[G[index][0]] = 1 - G[index][1]

# fragile.append(dic)

#print(fragile)

#name = list()

#value = list()

#for row in range(nrows-3):

# name.append(G[row][0])

# value.append(G[row][1])

#print(name)

#print(value)

**divThree.py**

import numpy as np

import pandas as pd

from fragile import fragile

G = np.array(fragile)

print(G)

a, b, c = np.random.randint(0, 10, 3)

#a = 2

#b = 6

#c = 8

#print(a,b,c)

print(G[a][0])

print(G[b][0])

print(G[c][0])

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

#print(G[a][1], G[b][1], G[c][1])

arr\_c = list()

arr\_c.append(G[a][1])

arr\_c.append(G[b][1])

arr\_c.append(G[c][1])

arr\_c = list(map(float, arr\_c))

last\_G1 = np.array([[1]])

last\_G2 = np.array([[1]])

last\_G3 = np.array([[1]])

curr\_G1 = np.array([[2]])

curr\_G2 = np.array([[2]])

curr\_G3 = np.array([[2]])

last\_ar0 = 1

last\_ar1 = 1

last\_ar2 = 1

curr\_ar0 = 0

curr\_ar1 = 0

curr\_ar2 = 0

while True:

G1 = []

G2 = []

G3 = []

#print('\*\*\*\*\*\*\*\*\*\*')

#print(last\_G1)

#print(curr\_G1)

#print(last\_G2)

#print(curr\_G2)

#print(last\_G3)

#print(curr\_G3)

#print('----------')

#print((last\_G1 == curr\_G1).all())

if last\_G1.shape == curr\_G1.shape and last\_G2.shape == curr\_G2.shape and \

last\_G3.shape == curr\_G3.shape and (last\_G1 == curr\_G1).all()==True and \

(last\_G2 == curr\_G2).all()==True and (last\_G3 == curr\_G3).all()==True:

print('bingo')

print(curr\_G1)

print(curr\_G2)

print(curr\_G3)

break

if last\_ar0 == curr\_ar0 and last\_ar1 == curr\_ar1 and last\_ar2 == curr\_ar2:

print('haha')

print(curr\_G1)

print(curr\_G2)

print(curr\_G3)

break

last\_G1 = np.copy(curr\_G1)

last\_G2 = np.copy(curr\_G2)

last\_G3 = np.copy(curr\_G3)

for index in range(len(G)):

x = np.abs(float(G[index][1])-arr\_c[0])

y = np.abs(float(G[index][1])-arr\_c[1])

z = np.abs(float(G[index][1])-arr\_c[2])

min\_v = min(x, y, z)

if min\_v == x:

G1.append(G[index])

elif min\_v == y:

G2.append(G[index])

elif min\_v == z:

G3.append(G[index])

G1 = np.array(G1)

G2 = np.array(G2)

G3 = np.array(G3)

G1x = list()

G2x = list()

G3x = list()

#print(G1.shape)

#print(G1)

for index in range(G1.shape[0]):

G1x.append(float(G1[index][1]))

for index in range(G2.shape[0]):

G2x.append(float(G2[index][1]))

for index in range(G3.shape[0]):

G3x.append(float(G3[index][1]))

#print(G1x)

arr\_c[0] = np.mean(G1x)

arr\_c[1] = np.mean(G2x)

arr\_c[2] = np.mean(G3x)

#print(arr\_c[0])

#print(arr\_c[1])

#print(arr\_c[2])

curr\_G1 = np.copy(G1)

curr\_G2 = np.copy(G2)

curr\_G3 = np.copy(G3)

curr\_ar0 = arr\_c[0]

curr\_ar1 = arr\_c[1]

curr\_ar2 = arr\_c[2]

#print('\*\*\*\*\*\*\*\*\*\*')

#print(last\_G1)

#print(curr\_G1)

#print(last\_G2)

#print(curr\_G2)

#print(last\_G3)

#print(curr\_G3)

#print('----------')

print('over')

**solvetime.py**

import numpy as np

import pandas as pd

data = pd.read\_excel('./resource/10yeardata.xlsx')

#print(data)

array = np.array(data.values)

#print(array)

f\_array = list()

for index in range(len(array)):

f\_array.append(array[index][1])

f\_array = np.array(f\_array)

#print(len(f\_array))

f1\_array = np.cumsum(f\_array)

#print(len(f1\_array), 'sdf')

g\_array = list()

for index in range(len(f1\_array)-1):

g\_array.append((f1\_array[index] + f1\_array[index + 1])/2)

#print(len(g\_array), 'asdf')

B\_array = list()

for index in range(len(g\_array)):

tmp = list()

tmp.append(-g\_array[index])

tmp.append(1)

B\_array.append(tmp)

B\_array = np.array(B\_array)

#print(len(B\_array))

Y\_array = list()

for index in range(1, len(f\_array)):

Y\_array.append(f\_array[index])

Y\_array = np.array(Y\_array)[np.newaxis,].T

#print(Y\_array.shape)

arr = np.mat(np.dot(B\_array.T, B\_array)).I

arr = np.dot(arr, B\_array.T)

a, b = np.dot(arr, Y\_array)

#print(a[0,0], b[0,0])

ba = b[0,0] / a[0,0]

f1\_array[0] = f\_array[0]

#print(f1\_array[0])

f1x\_array = list()

f1x\_array.append(f1\_array[0])

for index in range(18):

f1x\_array.append((f1\_array[0]-ba)\*np.exp(-a[0,0]\*index) + ba)

#print(f1x\_array)

f0x\_array = list()

f0x\_array.append(f\_array[0])

for index in range(1, 19):

# print(index)

f0x\_array.append(f1x\_array[index]-f1x\_array[index-1])

print(f0x\_array)

dx0\_array = list()

dx\_array = list()

for index in range(8):

dx0\_array.append(f\_array[index]-f0x\_array[index])

for index in range(8):

dx\_array.append(abs(dx0\_array[index]/f\_array[index]))

res = 0

for index in range(8):

res = res + dx\_array[index]

dx = res / 9

#print(dx)