北京航空航天大學
BEIHANG UNIVERSITY

2023—2024 学年　第 二 学期

《大学计算机基础（理科）》

大作业实验报告

班 级	237711	学 号	23377021
姓 名	何健	成 绩	

2024 年 6 月

题目三　简易插值拟合求解器

1　题目描述

插值与拟合是常用的数据处理手段，在科学计算中得到了广泛的应用。

设n是大于3的正整数a,b,c是实数关于$t_1, t_2, ..., t_n, t_{n+1}, t_{n+2}$的齐次线性方程组

$$\begin{cases} at_1 + bt_2 + ct_3 = 0 \\ at_2 + bt_3 + ct_4 = 0 \\ ... \\ at_{n-1} + bt_n + ct_{n+1} = 0 \\ at_n + bt_{n+1} + ct_{n+2} = 0 \end{cases}$$

以及关于t_{n+1}, t_{n+2},z的两个数表（如表1,详见Excel数据表附件）确定了一个二元函数z=f(t_1, t_2)。在本题中取n=10,a=1,b=-1,c=1。

表 1: 数表示例

t_{n+2}/z/t_{n+1}	-0.1251	-0.5756	-2.2577	-0.8147	...
-2.8313	0.0579949	0.0320723	0.0018421	0.0226652	...
-2.7001	0.0649102	0.0361016	0.0020696	0.0255959	...
-1.8984	0.273669	0.1525219	0.0081513	0.1080882	...
-2.748	0.0620372	0.34437	0.0019748	0.0243888	...
...	...	...	...	...	...

请设计实现一个简易插值拟合求解器，能够对三维欧式空间中的点集{(x,y,z)}进行插值拟合形成曲面。为简便起见，下面把t_1记为x，t_2记为y。

实验要求：

1.请使用Python提供的Pandas库完成以下工作：

(1)读入数据表附件“data.xlsx”；

(2)将两个数表拼接成一个数表；

(3)对数表做行列交换，要求t_{n+1}的取值栏从左至右升序，t_{n+2}的取值栏从上至下降序；

(4)找出数表中-0.5≤t_{n+1}≤0且0≤t_{n+2}≤2的部分，打印该新数表的行数和列数；

(5)导出新数表，格式自选(可以选择csv、excel等格式)。

2.请使用Python科学计算库numpy、scipy提供的求解线性方程组、插值、拟合的函数完成以下工作：
(1)设当$x = x_i, y = y_j$时,$t_{n+1} = u_p, t_{n+2}$=v_q.求集合$\{u_p\}_{1\leqslant p\leqslant p_0}$和$\{v_q\}_{1\leqslant q\leqslant q_0}$,并打印$p_0, q_0$的值，其中$x_i = 0.01i$, y_j=0.02j(0⩽i⩽200,0⩽j⩽150)；
(2)设x_i和y_j的定义如(1)，求f(x,y)($D = \{(x, y) \mid 0\leqslant x\leqslant 2, 0\leqslant y\leqslant 3\}$的近似表达式：

$$p(x, y) = \sum_{r=0}^{k}\sum_{s=0}^{k} c_{rs}x^r y^s$$

要求p(x,y)以最小的k值(记为k_{min})达到以下精度要求：

$$\sigma = \sum_{i=0}^{200}\sum_{j=0}^{150}[f(x_i, y_i) - p(x_i, y_i)]^2 \leqslant 1$$

并打印当k=1 , . . . , k_{min}时σ的取值以及k=k_{min}时的系数矩阵$[c_{rs}]$。

3.(1) 绘制平面热力图figure1，数据为实验要求1(4)中的数表$\{(t_{n+1}, t_{n+2}, z)\}$；
(2) 绘制平面散点图figure2，包括2个子图，子图1的数据为实验要求1(3)中的数表 $\{(t_{n+1}, t_{n+2}, z)\}$，子图2的数据为数表$\{(u_p, v_q, f(x_i, y_j))\}$，以观察插值效果；
(3) 绘制空间散点图figure3，首先画出符合实验要求2(2)中k=2时的曲面p(x,y)；其次将点$(x_i^*, y_j^*, f(x_i^*, y_j^*))$和点$(x_i^*, y_j^*, p(x_i^*, y_j^*))$在图上连接起来，以观察p(x,y)拟合f(x,y)的效果。要求打印数表$\{x_i^*, y_j^*, f(x_i^*, y_j^*), p(x_i^*, y_j*)\}$，其中$x_i^*$=0.25i,$y_j^*$=0.5j(0⩽i⩽8,0⩽j⩽6)。

2 总体设计方案

本作业实现的功能有：读入、排序、合并和筛选df数据表，将数据表写入xlsx工作表，解线性方程组，插值，拟合，绘制热力图、平面散点图和立体散点图。

本程序总共分为三大部分，分别是df数据表处理部分、插值和拟合部分、绘制图像部分，与实验要求中的第1、2、3题分别对应。

df数据表处理部分使用pandas库中的读入、排序、合并和筛选语法将给出的xlsx工作表读入df数据表，并对df数据表进行排序合并和筛选，最后输出筛选后的df数据表的行数和列数。

插值和拟合部分是本程序的重难点，分为三个模块，分别是数据处理模块、插值模块和拟合模块，数据处理模块主要用scipy库中的解线性方程组的语法处理数据，插值模块和拟合模块则主要使用了scipy库中的griddata进行插值，curve_fit进行拟合获得了插值后的二维数据列表f和拟合后的函数poly。

绘制图像部分分为三个模块，分别是热力图模块、平面散点图模块和立体散点图模块。本部分引入了matplotlib.pyplot和seaborn库，分别用seaborn，scatter绘制热力图和散点图。

3 设计思路和关键代码

本程序代码分为三个部分，每个部分都会给出程序代码，核心语法，对实验要求实现方法的详细分析和关键代码。

3.1 第一部分：

程序代码：

```
df1 = pd.read_excel('data.xlsx',sheet_name='Sheet1')
df2 = pd.read_excel('data.xlsx',sheet_name='Sheet2')#导入工作表数据
df3 = pd.concat([df1,df2],ignore_index=True)#合并两个dataframe
df3.sort_values(by=0,axis=1,ascending=True,inplace=True)#按列升序排列
df3.sort_values(by=0,ascending=False,inplace=True)#按行降序排列
column_names = df3.columns.tolist()#将列索引转换为列表
for i in range(1,len(column_names)):
    if column_names[i] >= -0.5:#寻找t_n+1>=0.5时的列索引的整数位置
       break

list_0 = []
column_names[0] = 't_n+2/t_n+1'
for j in column_names:
    list_0.append(str(j))#将列名称数据类型转换为字符串

df3.columns = list_0#改变列索引
df3 = df3.reset_index(drop=True)#重置行索引
select_df3 = df3[df3['t_n+2/t_n+1']>=0]#筛选出0=<t_n+2<=2的行
df3_0 = select_df3.iloc[:,[0]]#筛选出第一列
df3_1 = select_df3.iloc[:,i:len(column_names)]#筛选出-0.5=<t_n+1<=0的列
df3_2 = pd.concat([df3_0, df3_1], axis=1)#合并两个数表
print(f'行数: {df3_2.shape[0]}, 列数: {df3_2.shape[1]}')
df3_2 = df3_2.reset_index(drop=True)#重置行索引
print(df3_2)
df3_2.to_excel('data2.xlsx', index=False)#将dataframe写入工作表
```

核心语法一览：

```
pd.read_excel()#导入工作表
pd.concat([,])#合并两个dataframe
df.sort_values()#按行或按列排列
df.columns.tolist()#将列索引转换为列表
df.reset_index(drop=)#重置行索引
df[df['']>=0]#筛选行
df.iloc[:,]#按列索引的整数位置筛选列
df.to_excel()#将dataframe写入工作表
```

实验要求的实现方法及关键代码：

本部分的主要难点在于，给出的excel工作表转换为dataframe之后，列名称的数据类型为浮点数，会导致无法使用筛选语法来筛选df数据表的行和列。

本部分程序先使用pd.read_excel()导入工作表，再用pd.concat([,])合并两个df数据表，利用df.sort_values()将df数据表按行和列排序，接着通过一系列代码将列名称的数据类型转换为字符串，使df数据表的筛选语法能正常使用，再对df数表筛选，最后将新生成的df数表写入新的excel工作表。

本部分的关键模块为将列名称的数据类型转换为字符串再进行筛选，关键算法如下：

```
column_names = df3.columns.tolist()#将列索引转换为列表
list_0 = []
column_names[0] = 't_n+2/t_n+1'
for j in column_names:
    list_0.append(str(j))#运用for循环将列名称数据类型转换为字符串

df3.columns = list_0#改变列索引
df3 = df3.reset_index(drop=True)#重置行索引
select_df3 = df3[df3['t_n+2/t_n+1']>=0]#筛选出0=<t_n+2<=2的行
df3_0 = select_df3.iloc[:,[0]]#筛选出第一列
df3_1 = select_df3.iloc[:,i:len(column_names)]#筛选出-0.5=<t_n+1<=0的列
df3_2 = pd.concat([df3_0, df3_1], axis=1)#合并两个数表
```

df3_2即为实验要求的筛选后的数表，将行数和列数打印出来即完成实验要求。

3.2 第二部分：

程序代码：

```
x_k = []
y_l = []
for k in range(0,201):
    x_k.append(float(f'{0.01*k:.4f}'))

for l in range(0,151):
    y_l.append(float(f'{0.02*l:.4f}'))

u_p = []
v_q = []#不包含重复数据
u_p_1 = []
v_q_1 = []#包含重复数据
A = np.array([[1,-1,1,0,0,0,0,0,0,0,0,0],
```

```
                [0,1,-1,1,0,0,0,0,0,0,0,0],
                [0,0,1,-1,1,0,0,0,0,0,0,0],
                [0,0,0,1,-1,1,0,0,0,0,0,0],
                [0,0,0,0,1,-1,1,0,0,0,0,0],
                [0,0,0,0,0,1,-1,1,0,0,0,0],
                [0,0,0,0,0,0,1,-1,1,0,0,0],
                [0,0,0,0,0,0,0,1,-1,1,0,0],
                [0,0,0,0,0,0,0,0,1,-1,1,0],
                [0,0,0,0,0,0,0,0,0,1,-1,1]])#系数矩阵
rank_A = np.linalg.matrix_rank(A)#求出系数矩阵的秩为10
ns = null_space(A)#求出齐次线性方程组的基础解系
'''基础解系为ns=
[[-0.16895503  0.37164615]
 [-0.40633253  0.03950373]
 [-0.23737749 -0.33214243]
 [ 0.16895503 -0.37164615]
 [ 0.40633253 -0.03950373]
 [ 0.23737749  0.33214243]
 [-0.16895503  0.37164615]
 [-0.40633253  0.03950373]
 [-0.23737749 -0.33214243]
 [ 0.16895503 -0.37164615]
 [ 0.40633253 -0.03950373]
 [ 0.23737749  0.33214243]]'''
B = np.array([[ns[0][0],ns[0][1]],
              [ns[1][0],ns[1][1]]])
'''以上两行的作用是构造新的线性方程组：
k1*s11+k2*s12=x_k,
k1*s21+k2*s22=y_l,
其中s11=ns[0][0],s12=ns[0][1]，以此类推'''
xy2 = []#创造待插值的点集
for j1 in y_l:
    for i1 in x_k:
        C = np.array([[i1],
                      [j1]])
        D = np.linalg.solve(B,C)#求出k_1和k_2并保存在D中
        a = round(D[0][0]*ns[10][0] + D[1][0]*ns[10][1],4)#得到t_n+1
```

```
        u_p_1.append(a)
        if a not in u_p:
            u_p.append(a)

        b = round(D[0][0]*ns[11][0] + D[1][0]*ns[11][1],4)#得到t_n+2
        v_q_1.append(b)
        if b not in v_q:
            v_q.append(b)

        xy2.append([a,b])

p_0 = len(u_p)
q_0 = len(v_q)
print(f'p_0={p_0},q_0={q_0}')#打印p_0,q_0
column_names.pop(0)
t_n_1 = column_names
t_n_2 = df3['t_n+2/t_n+1'].tolist()
for i2 in t_n_1:
    i2 = round(i2,4)#将数字四舍五入到四位小数
for j2 in t_n_2:
    j2 = round(j2,4)#将数字四舍五入到四位小数

z = []#创建z的二维列表
values = []
for i4 in range(0,50):
    values = df3.iloc[i4].tolist()
    values.pop(0)
    z.append(values)

new_z = np.array(z).reshape(2500)#将z展开成一维数组
points = []
for i12 in t_n_2:
    for j12 in t_n_1:
        points.append([j12,i12])

points = np.array(points)
'''以下为插值'''
```

```
X2,Y2 = np.meshgrid(u_p,v_q)#创建网格
f2 = griddata(points,new_z,(X2,Y2),method='cubic')#进行插值
'''以下为拟合'''
f_3 = []#建立与每个点一一对应的函数值的列表
for i13 in xy2:
    d = u_p.index(i13[0])#找出每个点的横坐标在f2中的第二维索引
    e = v_q.index(i13[1])#找出每个点的纵坐标在f2中的第一维索引
    f_3.append(f2[e][d])

f_3 = np.array(f_3)
xy = []
for i8 in y_l:
    for j8 in x_k:
        xy.append([j8,i8])

xy = np.array(xy)#创建需要拟合的点的点集
popt = []#用于保存拟合函数的系数列表
sigma = []#用于保存误差值
def poly(order,arr,xy):
    z1 = 0
    x,y = xy[:,0],xy[:,1]#分解二维数组为x和y
    for i18 in range(0,order+1):
        for j18 in range(0,order+1):
            z1 += arr[i18*(order+1)+j18]*x**i18*y**j18#建立拟合函数
    return z1

def polynomial_func(xy,*coefficients):#coefficients为储存系数的列表
    order = int(len(coefficients)**(1/2)-1)#order为多项式阶数
    return poly(order,coefficients,xy)

for i11 in range(1,7):
    popt1,pcov1 = curve_fit(polynomial_func,xy,f_3,p0=[0]*((i11+1)**2))
    '''以上一行进行拟合，分别得到系数数组和协方差，p0的作用为初始化系数列表'''
    popt.append(np.round(popt1,2))
    p_1 = polynomial_func(xy,*popt1)#调用拟合函数得到拟合值
    sigma1 = 0#初始化误差值
    for i9 in range(0,len(p_1)):
```

```
        sigma1 += (f_3[i9]-p_1[i9])**2#计算误差值

    sigma.append(sigma1)
    print(f'k={i11}时,sigma={sigma1}')

c_rs = np.array(popt[-1]).reshape(i11+1,i11+1)#c_rs为k=k_min时的系数矩阵
print(f'k=k_min时,系数矩阵为c_rs=\n{c_rs}')
```

核心语法一览：

```
list.pop()#弹出列表中的某项
list.append()#将某项插入列表的末尾
np.array()#创建数组
np.linalg.matrix_rank()#求出系数矩阵的秩
ns = null_space(A)#求出齐次线性方程组的基础解系
np.linalg.solve(B,C)#求解非齐次线性方程组
round(x,4)#保留小数位数
len(list)#求列表长度
np.reshape()#重新塑造数组维数
X,Y = np.meshgrid(x,y)#创建网格
f = griddata(points,z,(X,Y),method='')#插值函数
list.index()#求出某项在列表中的索引
x,y = xy[:,0],xy[:,1]#分解二维数组
def poly(order,arr,xy)#定义拟合函数
def polynomial_func(xy,*coefficients)#定义拟合函数
popt1,pcov1 = curve_fit(polynomial_func, , ,p0=[])#进行拟合
p_1 = polynomial_func(xy,*popt1)#调用拟合函数得到拟合值
```

实验要求的实现方法及关键代码：

实验要求的实现方法为：首先创造x_k,y_l的列表，已知点集(x_k, y_l)共有30351个点。列出系数矩阵A，求出齐次线性方程组的基础解系ns，再构建新的线性方程组，利用ns和点集(x_k, y_l)求出u_p和v_q的列表，这两个列表的长度都为30351。去掉其中的重复项，得到长度分别为151和501的集合$\{u_p\}$和$\{v_q\}$，用集合$\{u_p\}$和$\{v_q\}$创建网格X,Y。接着将数表中的数据转换成2500组(t_{n+1}, t_{n+2})和2500个与之一一对应的准确值z。用griddata将2500组准确的(t_{n+1}, t_{n+2}, z)插值成网格上75651组(u_p, v_q, f)。再利用求列表索引的方法将30351组(x_k, y_l)与插值出的点(u_p, v_q)一一对应，进而找到一一对应的插值函数值f。至此，拟合所需的所有数据处理工作已经准备完毕，只需要写出拟合函数的算法再进行拟合，即可得到拟合函数p(x,y)，不同多项式阶数k对应的σ值以及k=k_{min}时的系数矩阵c_{rs}。

本部分的主要难点有三个部分：解线性方程组、利用列表索引将(x_k, y_l)和插值f一一对应、拟合函数的建立。以下为本部分的几个关键代码块：

代码块一：

```
A = np.array([[1,-1,1,0,0,0,0,0,0,0,0,0],
              [0,1,-1,1,0,0,0,0,0,0,0,0],
              [0,0,1,-1,1,0,0,0,0,0,0,0],
              [0,0,0,1,-1,1,0,0,0,0,0,0],
              [0,0,0,0,1,-1,1,0,0,0,0,0],
              [0,0,0,0,0,1,-1,1,0,0,0,0],
              [0,0,0,0,0,0,1,-1,1,0,0,0],
              [0,0,0,0,0,0,0,1,-1,1,0,0],
              [0,0,0,0,0,0,0,0,1,-1,1,0],
              [0,0,0,0,0,0,0,0,0,1,-1,1]])#系数矩阵
rank_A = np.linalg.matrix_rank(A)#求出系数矩阵的秩为10
ns = null_space(A)#求出齐次线性方程组的基础解系
'''基础解系为ns=
[[-0.16895503  0.37164615]
 [-0.40633253  0.039503 73]
 [-0.23737749 -0.33214243]
 [ 0.16895503 -0.37164615]
 [ 0.40633253 -0.039503 73]
 [ 0.23737749  0.33214243]
 [-0.16895503  0.37164615]
 [-0.40633253  0.039503 73]
 [-0.23737749 -0.33214243]
 [ 0.16895503 -0.37164615]
 [ 0.40633253 -0.039503 73]
 [ 0.23737749  0.33214243]]'''
B = np.array([[ns[0][0],ns[0][1]],
              [ns[1][0],ns[1][1]]])
'''以上两行的作用是构造新的线性方程组:
k1*s11+k2*s12=x_k,
k1*s21+k2*s22=y_l,
其中s11=ns[0][0],s12=ns[0][1], 以此类推'''
xy2 = []#创造待插值的点集
for j1 in y_l:
    for i1 in x_k:
        C = np.array([[i1],
                      [j1]])
        D = np.linalg.solve(B,C)#求出k_1和k_2并保存在D中
```

```
        a = round(D[0][0]*ns[10][0] + D[1][0]*ns[10][1],4)#得到t_n+1
        u_p_1.append(a)
        if a not in u_p:
            u_p.append(a)

        b = round(D[0][0]*ns[11][0] + D[1][0]*ns[11][1],4)#得到t_n+2
        v_q_1.append(b)
        if b not in v_q:
            v_q.append(b)

        xy2.append([a,b])
```

以上代码块的思路为列出系数矩阵，求出基础解系，利用for循环解线性方程组得到集合$\{u_p\}$和$\{v_q\}$，最后打印$\{u_p\}$和$\{v_q\}$的长度，得到实验要求中p_0和q_0的值。

代码块二：

```
f_3 = []#建立与每个点一一对应的函数值的列表
for i13 in xy2:
    d = u_p.index(i13[0])#找出每个点的横坐标在f2中的第二维索引
    e = v_q.index(i13[1])#找出每个点的纵坐标在f2中的第一维索引
    f_3.append(f2[e][d])
```

以上代码块虽然简短，但是需要清晰的逻辑思路，xy2是包含30351组(u_p, v_q)的点集，由于网格(X2,Y2)是由u_p和v_q形成的,所以(u_p, v_q)的横坐标在u_p中的索引对应f2这个二维列表中的第二维索引，(u_p, v_q)的纵坐标在v_q中的索引对应f2这个二维列表中的第一维索引，这样(u_p, v_q)就与f2中的函数值建立了一一对应关系，由于(x_k, y_l)与(u_p, v_q)也是一一对应关系，所以就建立了(x_k, y_l)与f2中的函数值的一一对应关系，为后来的拟合打下了基础。

代码块三：

```
def poly(order,arr,xy):
    z1 = 0
    x,y = xy[:,0],xy[:,1]#分解二维数组为x和y
    for i18 in range(0,order+1):
        for j18 in range(0,order+1):
            z1 += arr[i18*(order+1)+j18]*x**i18*y**j18#建立拟合函数
    return z1

def polynomial_func(xy,*coefficients):#coefficients为储存系数的列表
    order = int(len(coefficients)**(1/2)-1)#order为多项式阶数
    return poly(order,coefficients,xy)
```

```
for i11 in range(1,7):
    popt1,pcov1 = curve_fit(polynomial_func,xy,f_3,p0=[0]*((i11+1)**2))
    '''以上一行进行拟合，分别得到系数数组和协方差，p0的作用为初始化系数列表'''
    p_1 = polynomial_func(xy,*popt1)#调用拟合函数得到拟合值
```

以上代码块为本程序最困难最核心的一部分，先从第二个定义的函数polynomial_func(xy,*coefficients)看起，coefficients是长度为$(k+1)^2$的系数列表，其中k为拟合函数的多项式阶数，coefficients由以上代码块中倒数第三行中的p0=[0]*((i11+1)**2)进行初始化，接着通过order = int(len(coefficients)**(1/2)-1)反解出多项式的阶数，最后通过函数polynomial_func启动函数poly。poly中的z1行即为拟合函数，比如k=order=1时，待拟合的函数为$c_{00}+c_{01}y+c_{10}x+c_{11}xy$。该代码块中倒数第三行的popt1为拟合函数的系数数组，pcov1为该拟合函数的协方差，最后再用得到的popt1调用一次polynomial_func函数即可得到拟合值。

3.3 第三部分：

程序代码：

```
'''绘制热力图'''
z_1 = []
values_1 = []
for i5 in range(0,12):
    values_1 = df3_2.iloc[i5].tolist()
    values_1.pop(0)
    z_1.append(values_1)#从df3_2中创建z_1的二维列表

plt.figure(figsize=(10, 6))#创建画布
sns.heatmap(z_1,cmap='coolwarm',annot_kws={"size": 7},annot=True,fmt='.4f')
plt.title('figure1')
plt.show()
'''绘制平面散点图'''
'''由于scatter函数的x,y参数和颜色值c之间存在一一对应的关系,既然颜色值c(也就是f(x,y))有2500个值,
那么x,y也需要各有2500个,所以我们首先用for语句将t_n+1和t_n+2扩充为2500个值。'''
x_1 = []
y_1 = []
for i6 in range(0,50):
    for j6 in t_n_1:
        x_1.append(j6)#将t_n_1扩充到2500个值

for i7 in t_n_2:
    for j9 in range(0,50):
        y_1.append(i7)#将t_n_2扩充到2500个值
```

```
'''创建子图1'''
plt.subplot(1,2,1)
cmap = plt.get_cmap('jet')#设置颜色条渐变色
plt.scatter(x_1,y_1,c=new_z,s=0.5,marker='.',cmap=cmap)
plt.colorbar()#添加颜色条
'''创建子图2'''
plt.subplot(1,2,2)
camp = plt.get_cmap('jet')#设置颜色条渐变色
plt.scatter(u_p_1,v_q_1,c=f_3,marker='.',cmap=cmap)
plt.colorbar()#添加颜色条
plt.show()
'''绘制立体散点图'''
x_k_1 = []
y_l_1 = []
for k1 in range(0,9):
    x_k_1.append(float(f'{0.25*k1:.4f}'))#创造x_i^*的列表并保留四位小数

for l1 in range(0,7):
    y_l_1.append(float(f'{0.5*l1:.4f}'))#创造y_j^*的列表并保留四位小数

xy1 = []
for i10 in y_l_1:
    for j10 in x_k_1:
        xy1.append([j10,i10])#创造(x_i^*,y_j^*)的点集

xy1 = np.array(xy1)
'''以下两行为多项式阶数k=2时的拟合函数p(x,y),以绘制曲面p(x,y)'''
popt2,pcov2 = curve_fit(polynomial_func,xy,f_3,p0=[0]*9)
p_2 = polynomial_func(xy,*popt2)
'''重新定义一个函数,使xy1中的63个点可以通过该函数拟合函数值'''
def p(xy1,c_00,c_01,c_02,c_10,c_11,c_12,c_20,c_21,c_22):
    x,y = xy1[:,0], xy1[:,1]#分解二维数组为x和y
    return c_00+c_01*y+c_02*y**2+c_10*x+c_11*x*y+c_12*x*y**2+c_20*x**2+c_21*x**2*y+c_22*x**2*y**2

p_3 = p(xy1,*popt2)#得到由63个点(x_i^*,y_j^*)拟合出的函数值
xy3 = []
for j14 in y_l_1:
```

```
    for i14 in x_k_1:
        E = np.array([[i14],
                      [j14]])
        F = np.linalg.solve(B,E)#求出k_1和k_2并保存在F中
        f = round(F[0][0]*ns[10][0] + F[1][0]*ns[10][1],4)
        g = round(F[0][0]*ns[11][0] + F[1][0]*ns[11][1],4)
        xy3.append([f,g])

f_2 = []
for i15 in xy3:
    m = u_p.index(i15[0])
    n = v_q.index(i15[1])
    f_2.append(f2[n][m])
'''绘图'''
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(xy[:,0],xy[:,1],p_2,s=0.02,color='blue')#绘制平面p(x,y)
ax.scatter(xy1[:,0],xy1[:,1],f_2,s=7,color='red')#绘制散点
ax.view_init(azim=20,elev=10)#转换立体散点图的视角
'''以下代码是为了将63个散点与相同横纵坐标的拟合点连线'''
xy1_x = np.array(xy1[:,0].tolist()*2)#将xy1中的点的横坐标分别提取出来，再复制所有横坐标形成列表
xy1_y = np.array(xy1[:,1].tolist()*2)#将xy1中的点的纵坐标分别提取出来，再复制所有纵坐标形成列表
z1 = np.array(p_3.tolist()+f_2)#所有点对应的函数值，包括拟合和准确的函数值
df4 = pd.DataFrame({'xy_x':xy1_x,'xy_y':xy1_y,'z1':z1})
grouped = df4.groupby(['xy_x','xy_y'])
for name,group in grouped:
    if len(group) > 1:#只有当点的数量大于1时才绘制连接线
        z1_values = group['z1'].values#提取z坐标值
        x_value, y_value = name#由于x和y坐标相同可以直接取第一个点的x和y坐标
        ax.plot([x_value]*len(z1_values),[y_value]*len(z1_values),z1_values,color='green',lw=1)
        #绘制连接线

plt.title('figure3')
plt.show()
'''以下代码是为了打印实验要求中的数表'''
xy1_x1 = xy1[:,0].tolist()
xy1_y1 = xy1[:,1].tolist()
```

```
p2_2 = p_3.tolist()
df5 = pd.DataFrame({'x':xy1_x1,'y':xy1_y1,'f':f_2,'p':p2_2})
print(f'数表为:\n{df5}')
```

核心语法一览：

```
ply.figure(figsize=(,))#创建画布
sns.heatmap( ,cmap='',annot_kws={'size':},annot=,fmt='')#绘制热力图
plt.title('')#取名
plt.show()#显示图像
plt.subplot( , , ,)#创建子图
cmap = plt.get_cmap('')#设置颜色条渐变色
plt.scatter( , ,c=,s=,marker='',cmap=)#绘制散点图
plt.colorbar()#添加颜色条
ax = fig.add_subplot(111,projection='3d')#创建立体散点图
ax.scatter( , , ,s=,color='')#绘制立体散点图
pd.DataFrame()#创建df数表
grouped = df.groupby()#分类
ax.plot( , , ,color='',lw=)#绘制连接线
ax.view_init(azim=,elev=)#转换立体散点图的视角
```

实验要求的实现方法及关键代码：

df3_2为筛选过后的数表，用其中的数据z创建二维列表来绘制热力图。绘制实验要求中的平面散点图需要用plt.colorbar()添加颜色条，再将scatter中的第三个参数设置为c=new_z，将new_z中的值赋给颜色c，最后创建两个子图绘制平面散点图即可。

以下为关键代码：

```
'''以下两行为多项式阶数k=2时的拟合函数p(x,y),以绘制曲面p(x,y)'''
popt2,pcov2 = curve_fit(polynomial_func,xy,f_3,p0=[0]*9)
p_2 = polynomial_func(xy,*popt2)
'''重新定义一个函数,使xy1中的63个点可以通过该函数拟合函数值'''
def p(xy1,c_00,c_01,c_02,c_10,c_11,c_12,c_20,c_21,c_22):
    x,y = xy1[:,0], xy1[:,1]#分解二维数组为x和y
    return c_00+c_01*y+c_02*y**2+c_10*x+c_11*x*y+c_12*x*y**2+c_20*x**2+c_21*x**2*y+c_22*x**2*y**2

p_3 = p(xy1,*popt2)#得到由63个点(x_i^*,y_j^*)拟合出的函数值
xy3 = []
for j14 in y_l_1:
    for i14 in x_k_1:
        E = np.array([[i14],
```

```
                    [j14]])
        F = np.linalg.solve(B,E)#求出k_1和k_2并保存在F中
        f = round(F[0][0]*ns[10][0] + F[1][0]*ns[10][1],4)
        g = round(F[0][0]*ns[11][0] + F[1][0]*ns[11][1],4)
        xy3.append([f,g])

f_2 = []
for i15 in xy3:
    m = u_p.index(i15[0])
    n = v_q.index(i15[1])
    f_2.append(f2[n][m])
'''绘图'''
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(xy[:,0],xy[:,1],p_2,s=0.02,color='blue')#绘制平面p(x,y)
ax.scatter(xy1[:,0],xy1[:,1],f_2,s=7,color='red')#绘制散点
ax.view_init(azim=20,elev=10)#转换立体散点图的视角
'''以下代码是为了将63个散点与相同横纵坐标的拟合点连线'''
xy1_x = np.array(xy1[:,0].tolist()*2)#将xy1中的点的横坐标分别提取出来，再复制所有横坐标形成列表
xy1_y = np.array(xy1[:,1].tolist()*2)#将xy1中的点的纵坐标分别提取出来，再复制所有纵坐标形成列表
z1 = np.array(p_3.tolist()+f_2)#所有点对应的函数值，包括拟合和准确的函数值
df4 = pd.DataFrame({'xy_x':xy1_x,'xy_y':xy1_y,'z1':z1})
grouped = df4.groupby(['xy_x','xy_y'])
for name,group in grouped:
    if len(group) > 1:#只有当点的数量大于1时才绘制连接线
        z1_values = group['z1'].values#提取z坐标值
        x_value, y_value = name#由于x和y坐标相同可以直接取第一个点的x和y坐标
        ax.plot([x_value]*len(z1_values),[y_value]*len(z1_values),z1_values,color='green',lw=1)
        #绘制连接线

plt.title('figure3')
plt.show()
```

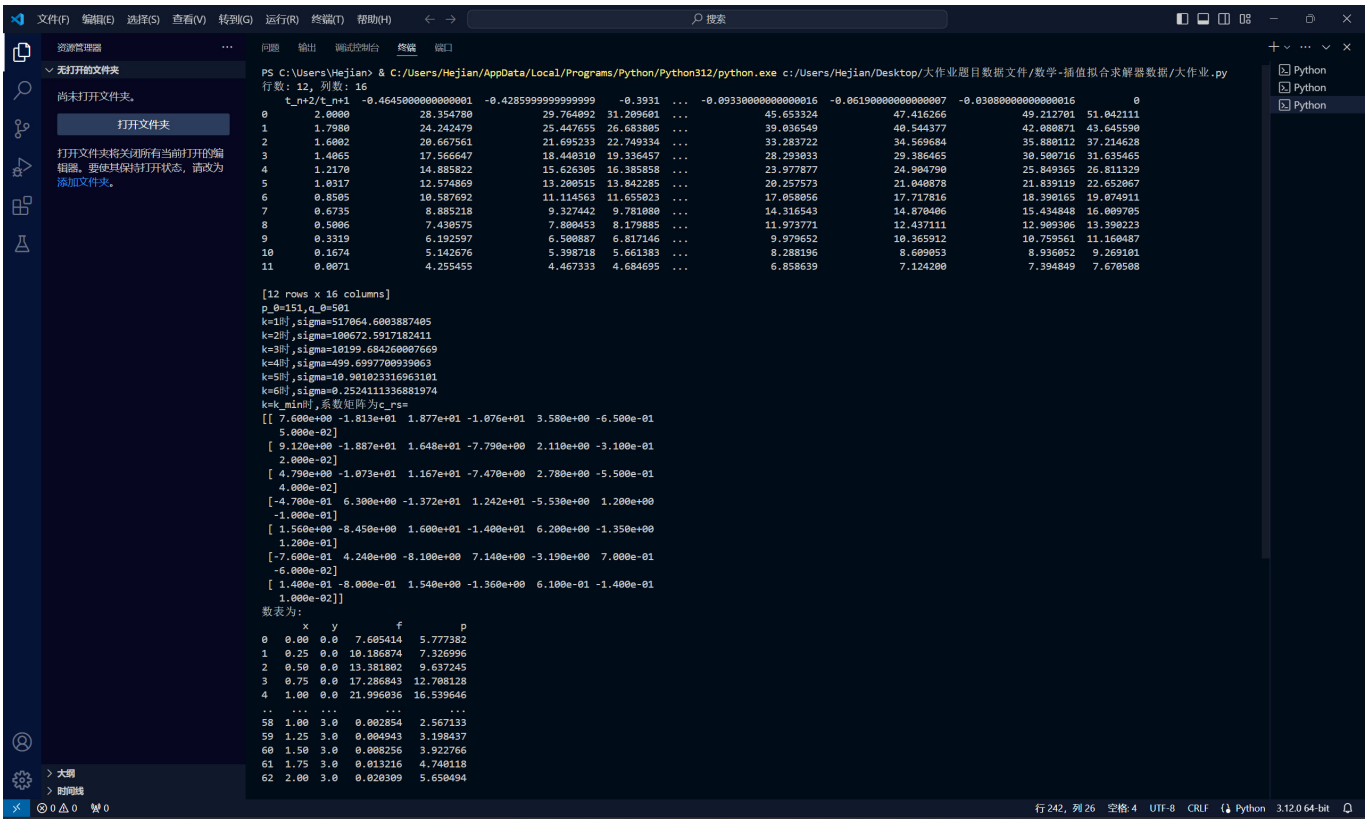
绘制立体散点图需要先进行数据处理，首先创建点集xy1=(x_i^*, y_j^*),解线性方程组，得到与xy1中的点一一对应的点集xy3，再通过上文求列表索引的方法，建立xy3中的点与f中的函数值的一一对应关系，从而建立(x_i^*, y_j^*)与f中的函数值的一一对应关系，这一步骤是为了绘制立体散点图中的63个准确点(x_i^*, y_j^*, f)。接着调用上文的函数并设置多项式阶数k=2，得到k=2时的曲面p(x,y)和63个点(x_i^*, y_j^*)的拟合函数值列表p_3。最后在同一个散点图中绘制出曲面p(x,y)和63个准确点(x_i^*, y_j^*, f)即可。

将具有相同横纵坐标的点连线需要使用df数表，先将xy1中的点的横纵坐标分别提取到两个列表xy1_x和xy1_y中，再将xy1_x和xy1_y中的数据全部复制一遍，这样做是为了在之后的df数表中分类时可以找到有相同横纵坐标的点，再将拟合出来的和插值出来的函数值放在同一个列表z1里，对xy1_x，xy1_y和z1建立df数表，分类得到有相同横纵坐标的点，最后用ax.plot()绘制连接线。

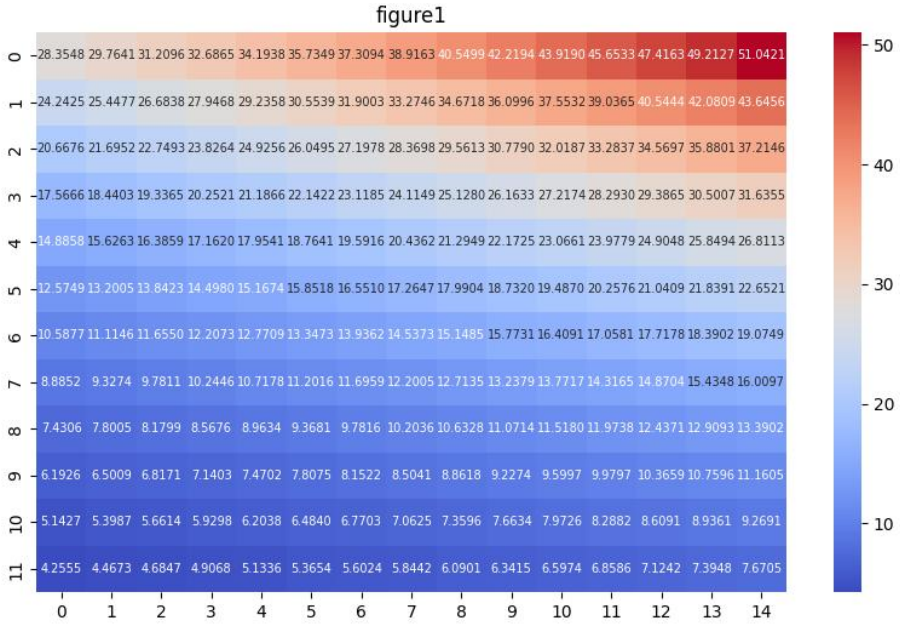
至此，所有实验要求全部完成。

4 程序运行效果

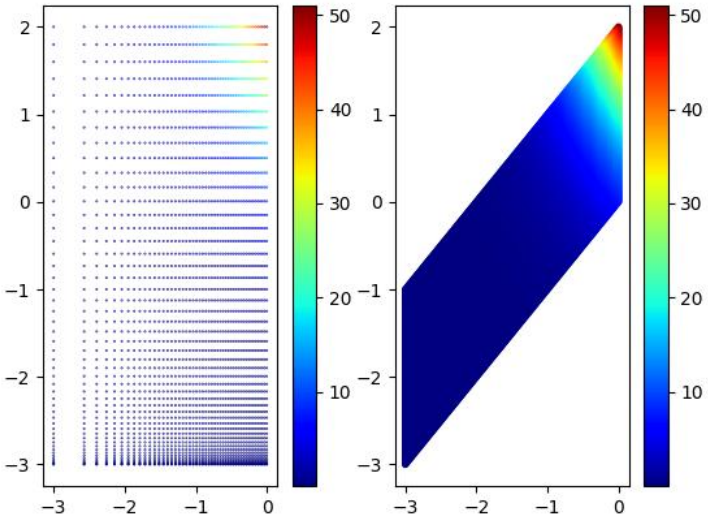
以下为程序运行结果截图及说明：



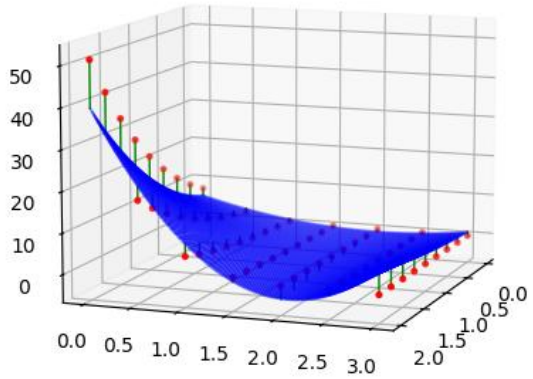
如图，最上方的数表为实验要求1.(4)中的新数表，也是本程序中的df3_2，数表上方即为该数表的行数和列数。该数表下方分别是集合$\{u_p\}$和$\{v_q\}$中的元素个数p_0和q_0。接下来是拟合出的函数多项式阶数k=1~6时对应的误差值σ以及k=k_{min}时的系数矩阵c_{rs}。最后是要求打印的数表$\{x_i^*, y_j^*, f(x_i^*, y_j^*), p(x_i^*, y_j*)\}$。



以上为要求绘制的热力图。



以上为要求绘制的平面散点图。



以上为要求绘制的立体散点图。

5 创新之处

本程序采用了一些课堂介绍之外的创新方法解决问题。

1.求解线性方程组时先求出系数矩阵的基础解系，再创建新的线性方程组求解得到各个点的横纵坐标值。

2.运用了griddata函数进行插值。

3.通过两个函数polynomial_func和poly相互作用来建立拟合多项式，并用curve_fit进行拟合。

4.通过添加颜色条来绘制平面散点图。

5.通过创建数表的方式绘制立体散点图上的连接线。

6 实验总结

1、在本实验中你遇到了哪些问题？是如何解决的？

答：本实验中遇到的问题和解决方案如下列举：

(1).由excel导入的工作表转换成df数表后，其列索引的数据类型是浮点数，导致df数表的筛选语法无法正常使用。解决方案是先提取列索引到一个列表中，将列索引的数据类型改为字符串后，再重置df数表的列索引。

(2).拟合的精度不够高，很长一段时间内我的程序运行出来k=15时才做到$\sigma \leqslant 1$，后来改进了插值的方式，由对(x_k, y_l)插值改为对(u_p, v_q)插值，解决了拟合精度不够高的问题。这也是本次大作业遇到的最大的问题。

2、有何收获和体会？你认为大作业题目还可以从哪些角度出题？请简要说明你的思路，谢谢！

答：对整个学期所学的乃至在课外学习的非常多的语法和库的运用都有了更高的熟练度，体会到了程序员的艰辛；可以对比不同拟合方式的精度，来选择更好地拟合函数。

7 课程学习总结

1、课程收获和难点分析

通过学习本课程你收获了什么？你认为现有内容中的难点在于哪些章节？

答：我学会了很多语法和程序设计思路；难点集中于运用matplotlib绘图、插值和拟合，科学计算和动态规划、贪心法等程序设计算法。

2、课程评价

答：建议课堂上多进行实例讲解而不是讲解语法，通过老师现场敲代码并展示程序运行效果的方式上课，并且可以对实验课上的题目进行讲解，在上到动态规划、贪心法等算法时重点进行讲解。实验内容建议不要夹带难以理解的私货。期中和期末考试建议按难度梯度设置题目顺序。

3、教师授课评价

答：教学态度不错，教学方式有提升空间。

4、助教评价

答：助教工作态度很好，也很有责任心，对我提出的问题热心解答，我很感谢谭浩君助教对我的帮助，同时建议一些助教出题时不要夹带难以理解的私货。

5、课程进一步改进建议

答：建议已在课程评价中给出。

8 主要参考资料和用到的库或工具

以下为本程序中对库的引用情况：

```
import pandas as pd
import numpy as np
from scipy.linalg import null_space
from scipy.interpolate import griddata
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.optimize import curve_fit
```

本次大作业运用的工具为python，spyder以及vscode，本实验报告的排版工具为Latex，主要参考资料为本python课程的课件以及一些代码网站。

致谢：谭浩君助教