第七章

插值与拟合

给定一组数据，需要求出不在此数据里几个数据点，可以用插值和拟合方法解决此问题：

求一个简单的函数作为要考察数据或复杂函数的近似，即求一条曲线（或曲面），通过此函数的值求得。如果所求的函数通过所有给定有限个数据点，这就是插值。如果不需要所求的函数通过所有数据点，而是需要此函数反映数据整体的变化态势，得到的近似函数，这就是曲线拟合。

目录 CONTENTS

实际应用中经常遇到如下问题:

已知某个未知函数$y=f(x)$的一系列数据点$(x_i,y_i)(i=0,1,\cdots,n)$，不妨设$x_0<x_1<\cdots<x_n$，但自变量$x$的其他值对应的函数值是未知的。希望通过这些数据点，得到函数$f(x)$的近似解析表达式，以便求出自变量x的值。

从某一函数类$\{P(x)\}$中，选出一个函数$P(x)$使得

$$P(x_i)=y_i(i=0,1,\cdots,n) \tag{7.0}$$

成立，则就以$P(x)$作为$f(x)$的近似函数，并且把函数$P(x)$称为插值函数，通常$x_0,x_1,\cdots,x_n$称为插值节点，$\{P(x)\}$称为插值函数类，(7.0)式称为插值条件，$f(x)$称为被插值函数。

插值函数类的取法有很多，可以是代数多项式，也可以是三角函数多项式或有理函数。由于代数多项式最简单，所以常用多项式作为插值函数类，以此来近似表达一些复杂的函数。

一维插值方法有很多，这里介绍一维 Lagrange 插值、牛顿插值、样条插值分段线性插值和分段二次插值。二维插值仅介绍二维样条插值的思想。

7.1.1 插值方法

1.Lagrange 插值

设有 n+1 个数据$(x_i, y_i)(i=0,1,\cdots,n)$，其中节点$x_0,x_1,\cdots,x_n$互异，则称满足插值条件（7.0）的 n 次多项式

$$P(x)=\sum_{i=0}^{n} l_i(x) y_i \tag{7.1}$$

为 n 次 Lagrange 插值多项式，其中$l_i(x)=\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}$称为 n 次 Lagrange 插值基函数。

例 7.1　编写函数 Lag_intp(x, y, x0)，实现拉格朗日插值。其中，x 和 y 是两个具有相同长度的 NumPy 数组。

```
#程序文件名 Pfun7_1.py
def h(x,y,a):
    s=0.0
    for i in range(len(y)):
        t=y[i]
        for j in range(len(y)):
            if i !=j:
                t*=(a-x[j])/(x[i]-x[j])
        s +=t
    return s
```

插值的误差估计

设$P(x)$是$f(x)$的n 次Lagrange多项式，有以下误差估计

$$|f(x) - P(x)| \leq \frac{M_n}{(n+1)!} \max |(x - x_0)(x - x_1) \cdots (x - x_n)|,$$

其中$M_n = \max |f^{(n+1)}(x)|.$

如果函数$f(x)$ 在区间$[a, b]$ 上有任意阶导数，并且存在与n无关的常数M 使得$\max_{a \leq x \leq b} |f^{(n+1)}(x)| \leq M$, 则有

$$\max_{a \leq x \leq b} |f(x) - P(x)| \leq \frac{M}{(n+1)!}(b-a)^{n+1} \longrightarrow 0,$$

这表明任一点处的插值的误差随插值节点的增多而减少。

2.分段线性插值

并非插值多项式次数越高误差越小，减小插值误差有效方法是分段插值方法。

分段多项式插值就是把插值节点分成几段（如 n 段），分别对每一段根据插值条件求相应插值多项式，最后得到一个分段（共n段）多项式$P(x)$，此分段多项式$P(x)$也满足插值条件。

设有数据$(x_i, y_i)(i=0,1,\cdots,n)$，$x_0 < x_1 < \cdots < x_n$，取$[x_i, x_{i+1}]$，由

$$P_1(x_i)=y_i, P_1(x_{i+1})=y_{i+1},$$

作线性插值多项式$P_1(x)$

$$P_1(x)=\frac{x-x_i}{x_{i+1}-x_i}y_{i+1}+\frac{x-x_{i+1}}{x_i-x_{i+1}}y_i,\quad x\in[x_i,x_{i+1}], i=0,1,\cdots,n-1. \qquad (7.2)$$

则在区间$[x_0, x_n]$上 $P_1(x)$就是分段一次多项式插值，几何上就是用折线代替曲线$y=f(x)$，也称折线插值或分段线性插值。

设$h=\max(x_{i+1}-x_i)$，分段线性插值的误差估计

$$\max_{a\le x\le b}|f(x)-P_1(x)| \le \frac{h^2}{8}\max_{a\le x\le b}|f''(x)|.$$

3.分段二次插值

设有数据$(x_i, y_i), i = 0, 1, \cdots, 2n, x_i < x_{i+1}$，在$[x_{2i}, x_{2i+2}]$

上做二次多项式

$$P_2(x) = \frac{(x - x_{2i+1})(x - x_{2i+2})}{(x_{2i} - x_{2i+1})(x_{2i} - x_{2i+2})} y_{2i} + \frac{(x - x_{2i})(x - x_{2i+2})}{(x_{2i+1} - x_{2i})(x_{2i+1} - x_{2i+2})} y_{2i+1}$$

$$+ \frac{(x - x_{2i})(x - x_{2i+1})}{(x_{2i+2} - x_{2i})(x_{2i+2} - x_{2i+1})} y_{2i+2},$$

其中$x \in [x_{2i}, x_{2i+2}]$，$i = 0, 1, 2, \cdots, n-1$。

在区间$[x_0,x_{2n}]$上 $P_2(x)$是一个分段 2 次多项式，在几何上它是分段抛物线代替曲线$y=f(x)$，也称分段抛物线插值。

分段抛物线插值具有误差估计

$$\max_{a\le x\le b}|f(x)-P_2(x)| \le \frac{h^3}{6}\max_{a\le x\le b}|f'''(x)|.$$

4.牛顿插值

(1) 函数的差分。

设有函数 $f(x)$ 以及等距节点 $x_i = x_0 + ih(i = 0,1,\cdots,n)$，步长 h 为常数，$f_i = f(x_i)$ 。称相邻两个节点 x_i, x_{i+1} 处的函数值的增量 $f_{i+1} - f_i(i = 0,1,\cdots,n-1)$ 为函数 $f(x)$ 在点 x_i 处以 h 为步长的一阶前向差分，记为 Δf_i，即

$$\Delta f_i = f_{i+1} - f_i, \quad (i = 0,1,\cdots,n-1).$$

两个一阶向前差分的差分，称为二阶前向差分，即

$$\Delta^2 f_i = \Delta f_{i+1} - \Delta f_i, \quad (i = 0,1,\cdots,n-2)$$

类似地可以定义 k（>2）阶向前差分，它们简称高阶差分。

一般地，归纳定义 $f(x)$ 的 m 阶前向差分 $\Delta^m f(x)$ 如下：

①$\Delta^0 f(x) = f(x)$；

②$\Delta^m f(x) = \Delta^{m-1} f(x+h) - \Delta^{m-1} f(x)$。

例 7.2　下面的函数递归地计算$\Delta^k f(x)$。

```
#程序文件名 Pfun7_2.py
def diff_forward(f, k, h, x):
    if k<=0: return f(x)
    else: return diff_forward(f, k-1, h, x+h) - diff_forward(f, k-1, h, x)
```

（2）函数的差商

设有函数 $f(x)$ 及一系列相异的节点 $x_0 < x_1 < \cdots < x_n$，则称 $\dfrac{f(x_i)-f(x_j)}{x_i-x_j}(i \neq j)$ 为函数 $f(x)$ 关于节点 x_i, x_j 的一阶差商，记为 $f[x_i, x_j]$，即

$$f[x_i, x_j] = \frac{f(x_i)-f(x_j)}{x_i-x_j}.$$

称一阶差商的差商

$$\frac{f[x_i,x_j]-f[x_j,x_k]}{x_i-x_k}$$

为 $f(x)$ 关于点 x_i,x_j,x_k 的二阶差商，记为 $f[x_i,x_j,x_k]$。一般地，称

$$\frac{f[x_0,x_1,\cdots,x_{k-1}]-f[x_1,x_2,\cdots,x_k]}{x_0-x_k}$$

为 $f(x)$ 关于点 $x_0,x_1,\cdots,x_k$ 的 k 阶差商，记为

$$f[x_0,x_1,\cdots,x_k]=\frac{f[x_0,x_1,\cdots,x_{k-1}]-f[x_1,x_2,\cdots,x_k]}{x_0-x_k}.$$

例 7.3　下面的函数用来递归计算 $f(x)$在给定点上的相应差商。

#程序文件名 Pfun7_3.py

""""计算 n 阶差商 f[x0, x1, x2 ... xn]

输入参数：xi 为所有插值节点的数组

输入参数：fi 为所有插值节点函数值的数组

返回值： 返回 x0,x1,…,xi 的 i 阶差商(i 为 xi 长度减 1)""""

```
def diff_quo(xi=[], fi=[]):
    if len(xi)>2 and len(fi)>2:
        return(diff_quo(xi[:len(xi)-1],fi[:len(fi)-1])-
            diff_quo(xi[1:len(xi)],fi[1:len(fi)]))\
                /float(xi[0]-xi[-1])   #续行
    return (fi[0]- fi[1])/float(xi[0]-xi[1])
```

(3）Newton 插值多项式

由于$y = f(x)$关于两节点x_0, x_1的线性插值多项式为

$$N_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0),$$

可将其表示成$N_1(x) = f(x_0) + (x - x_0) f[x_0, x_1]$，称为一次 Newton 插值多项式。

一般地，由各阶差商的定义，依次可得

$$f(x)=f(x_0)+(x-x_0)f[x,x_0],$$

$$f[x,x_0]=f[x_0,x_1]+(x-x_1)f[x,x_0,x_1],$$

$$f[x,x_0,x_1]=f[x_0,x_1,x_2]+(x-x_2)f[x,x_0,x_1,x_2],$$

$$\cdots\cdots$$

$$f[x,x_0,\cdots,x_{n-1}]=f[x_0,x_1,\cdots,x_n]+(x-x_n)f[x,x_0,\cdots,x_n],$$

将以上各式分别乘以

$$1,\ (x-x_0),\ (x-x_0)(x-x_1),\ \cdots,\ (x-x_0)(x-x_1)\cdots(x-x_{n-1}),$$

然后相加并消去两边相等的部分，即得

$$f(x)=f(x_0)+(x-x_0)f[x_0,x_1]+\cdots+(x-x_0)(x-x_1)\cdots(x-x_{n-1})f[x_0,x_1,\cdots,x_n]$$
$$+(x-x_0)(x-x_1)\cdots(x-x_n)f[x,x_0,x_1,\cdots,x_n],$$

记

$$N_n(x)=f(x_0)+(x-x_0)f[x_0,x_1]+\cdots+(x-x_0)(x-x_1)\cdots(x-x_{n-1})f[x_0,x_1,\cdots,x_n$$

$$R_n(x)=(x-x_0)(x-x_1)\cdots(x-x_n)f[x,x_0,x_1,\cdots,x_n],$$

显然，$N_n(x)$是至多n次的多项式，且满足插值条件，因而它是$f(x)$的n次插值多项式。这种形式的插值多项式称为 Newton 插值多项式。$R_n(x)$称为 Newton 插值余项。

函数系1，$(x-x_0)$，$(x-x_0)(x-x_1)$，$\cdots$，$(x-x_0)(x-x_1)\cdots(x-x_{n-1})$，是 Newton 插值多项式的插值基函数。

Newton 插值的优点是：每增加一个节点，插值多项式只增加一项，即

$$N_{n+1}(x)=N_n(x)+(x-x_0)\cdots(x-x_n)f[x_0,x_1,\cdots,x_{n+1}],$$

因而便于递推运算，而对 Lagrange 插值，增加节点时必须重新建立插值多项式，之前结果无法利用，因此 Newton 插值的计算量小于 Lagrange 插值。

牛顿插值的 Python 函数类似前面的 Lagrange 插值函数，同学们试试自己编写。

5.样条插值

样条（spline）是一根富有弹性的细木条或细金属条，在过去是工程设计中的一种绘图工具，绘图员利用它把已知点连接成一条光滑曲线（称为样条曲线），并使连接点处有连续曲率。在数学上可以证明这个曲线是分段 3 次多项式，称为三次样条函数。

许多工程技术中提出的计算问题对插值函数的光滑性有较高要求，如飞机的机翼外形，内燃机的进气门、排气门的凸轮曲线，都要求曲线不仅连续，而且具有连续的曲率，由此提出样条函数问题。

数学上将具有一定光滑性的分段多项式称为样条函数。具体地讲，给定区间$[a,b]$的一个分划

$$\Delta: a = x_0 < x_1 < \cdots < x_n = b.$$

如果函数$S(x)$满足：

（1）在每个小区间$[x_i, x_{i+1}]$ $(i = 0,1,\cdots,n-1)$上是m次多项式；

（2）在区间$[a,b]$上具有$m-1$阶连续导数；

则称$S(x)$为关于分划Δ的m次样条函数，其图形为m次样条曲线。显然，折线是一次样条曲线。

把样条函数作为插值函数类进行插值，称为样条插值，这里介绍三次样条插值。

给定函数 $y=f(x)$ 在区间 $[a,b]$ 上的 $n+1$ 个节点的值 $y_i=f(x_i)$ $(i=0,1,\cdots,n)$，计算插值函数 $S(x)$，使得 $S(x)$ 为分段三次多项式，在区间 $[a,b]$ 上 2 阶连续可导，且 $S(x_i)=y_i(i=0,1,\cdots,n)$。记

$$S(x)=\begin{cases} S_0(x)=a_0x^3+b_0x^2+c_0x+d_0, x\in[x_0,x_1], \\ S_1(x)=a_1x^3+b_1x^2+c_1x+d_1, x\in[x_1,x_2], \\ \cdots\cdots, \\ S_{n-1}(x)=a_{n-1}x^3+b_{n-1}x^2+c_{n-1}x+d_{n-1}, x\in[x_{n-1},x_n], \end{cases}$$

其中 a_i,b_i,c_i,d_i 为待定常数，共 $4n$ 个.

为了求出待定系数，利用样条函数的性质和插值条件得到$4n-2$个方程

$$\begin{cases} S(x_i)=y_i, \quad i=0,1,\cdots,n, \\ S_i(x_{i+1})=S_{i+1}(x_{i+1}), \quad i=0,1,\cdots,n-2, \\ S_i'(x_{i+1})=S_{i+1}'(x_{i+1}), \quad i=0,1,\cdots,n-2, \\ S_i''(x_{i+1})=S_{i+1}''(x_{i+1}), \quad i=0,1,\cdots,n-2. \end{cases}$$

要求出待定系数，还差两个条件，因此考虑边界条件。边界条件有三种类型：

（1）第一型：$S'(a)=y_0'$，$S'(b)=y_n'$；

（2）第二型：$S''(a)=y_0''$，$S''(b)=y_n''$；

（3）第三型（周期型）：$S'(a+0)=S'(b-0)$，$S''(a+0)=S''(b-0)$。

三次样条插值具有以下误差估计:

设被插值函数$f \in C^4[a,b]$, $S(x)$ 为满足第一型或第二型边界条件的三次样条插值样条函数，则在插值区间上有估计式

$$\|f^{(k)} - S^{(k)}\|_\infty \le c_k h^{4-k} \|f^{(4)}\|_\infty,\ k = 0, 1, 2,$$

其中c_k是常数, $h = \max\limits_{0 \le i \le n-1} h_i$, $h_i = x_{i+1} - x_i$.

6.二维数据的双三次样条插值

对于二维数据的插值，首先要考虑两个问题：一是二维区域是任意区域还是规则区域；二是给定的数据是有规律分布的还是散乱、随机分布的。

第一个问题比较容易处理，只需将不规则区域划分为规则区域或扩充为规则区域即可。对于第二个问题，当给定的数据是有规律分布时，方法较多也较成熟；而给定的数据是散乱、随机分布时，没有固定的方法，但一般的处理思想是：从给定的数据出发，依据一定的规律恢复出规则分布点上的数据，转化为数据分布有规律的情形来处理。

当给定的二维数据在规则区域上有规律分布时，通常用双三次样条插值方法，其基本思想是: 给定未知函数$z = f(x,y)$的二维观测数据如表 7.1 所示，对x轴和y轴的做分割:

$$\Delta x : x_1 < x_2 < \cdots < x_m, \quad \Delta y : y_1 < y_2 < \cdots < y_n$$

可以导出xy平面上矩形区域R的一个矩形网格分割$\Delta : \Delta x \times \Delta y$，如图 7.1 所示。

表 7.1　二维规则数据

	y_1	y_2	$\cdots$	y_n
x_1	z_{11}	z_{12}	$\cdots$	z_{1n}
x_2	z_{21}	z_{22}	$\cdots$	z_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_m	z_{m1}	z_{m2}	$\cdots$	z_{mn}

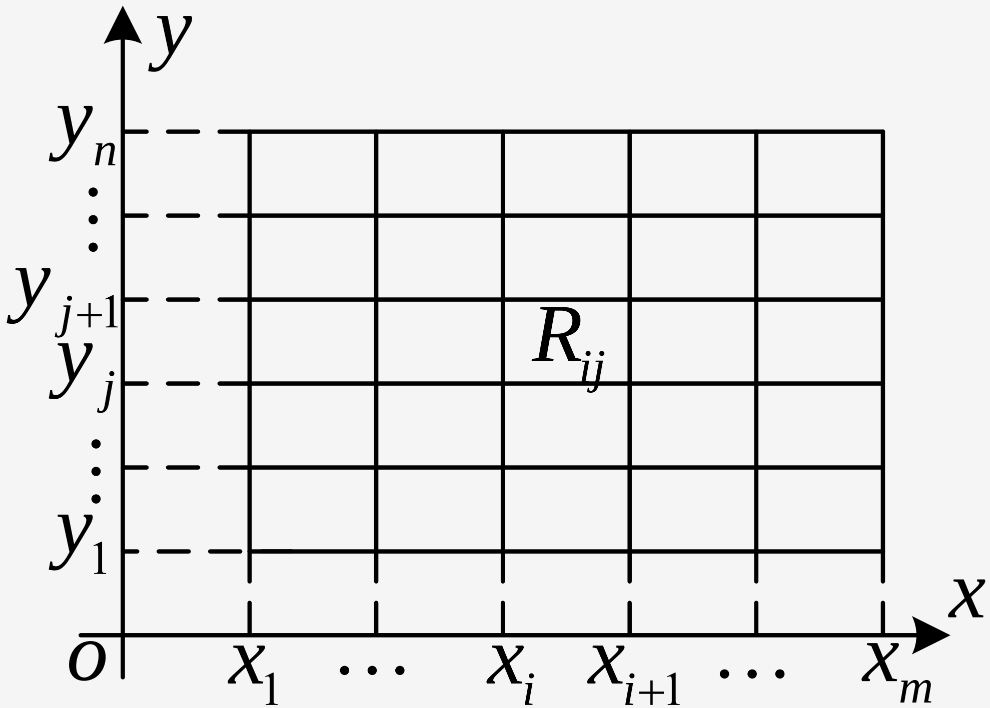

图 7.1　矩形网格分割Δ

如果令$R_{ij}:[x_i,x_{i+1}]\times[y_j,y_{j+1}], i=1,2,\cdots,m-1; j=1,2,\cdots,n-1$，则所谓双三次样条插值，就是求一个关于$x$和$y$都是三次的多项式$S(x,y)$，使其满足

（1）插值条件：$S(x_i,y_j)=z_{ij}, i=1,2,\cdots,m, j=1,2,\cdots,n$;

(2) 在整个R上，函数$S(x,y)$的偏导数$\dfrac{\partial^{\alpha+\beta}S(x,y)}{\partial x^{\alpha}\partial y^{\beta}}(\alpha,\beta=0,1,2)$都是连续的;

则称多项式$S(x,y)$为双三次样条插值函数，其插值公式为

$$S(x,y)=\sum_{k=0}^{3}\sum_{l=0}^{3}a_{kl}^{ij}(x-x_i)^k(y-y_j)^l,(x,y)\in R_{ij}. \tag{7.4}$$

实际上，双三次样条插值函数是由两个一维三次样条插值函数作直积产生的。对任意固定的$y_0 \in [y_1, y_n]$，$S(x, y_0)$是关于x的三次样条函数；同理，对任意固定的$x_0 \in [x_1, x_m]$，$S(x_0, y)$是关于y的三次样条函数。

7.1.2 用Python求解插值问题

scipy.interpolate 模块有一维插值函数 interp1d，二维插值函数 interp2d，多维插值函数 interpn、interpnd。

interp1d 的基本调用格式为

interp1d(x, y, kind='linear')

其中 kind 的取值是字符串，指明插值方法，kind 的取值可以为：linear', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic'等，这里的'zero', 'slinear', 'quadratic' and 'cubic'分别指的是 0 阶、1 阶、2 阶和 3 阶样条插值。

1.一维插值

例 7.4　在一天 24 小时内，从零点开始每间隔 2 小时测得的环境温度（摄氏度）如表 7.2 所示。分别进行分段线性插值和三次样条插值，并画出插值曲线。

表 7.2　24 小时环境温度数据

时间	0	2	4	6	8	10	12	14	16	18	20	22	24
温度	12	9	9	10	18	24	28	27	25	20	18	15	13

```
#程序文件名 Pex7_4.py
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
x=np.arange(0,25,2)
y=np.array([12, 9, 9, 10, 18, 24, 28, 27, 25, 20, 18, 15, 13])
xnew=np.linspace(0, 24, 500)    #插值点
f1=interp1d(x, y); y1=f1(xnew);
f2=interp1d(x, y,'cubic'); y2=f2(xnew)
plt.rc('font',size=16); plt.rc('font',family='SimHei')
```

```
plt.subplot(121), plt.plot(xnew, y1); plt.xlabel("（A）分段线性插值")
plt.subplot(122); plt.plot(xnew, y2); plt.xlabel("（B）三次样条插值")
plt.savefig("figure7_4.png", dpi=500); plt.show()
```

2.二维网格节点插值

例 7.5　已知平面区域$0 \le x \le 1400, 0 \le y \le 1200$的高程数据见表 7.3(单位：m)。求该区域地表面积的近似值，并用插值数据画出该区域的等高线图和三维表面图。

7.1插值

表 7.3 高程数据表

1200	1350	1370	1390	1400	1410	960	940	880	800	690	570	430	290	210	150
1100	1370	1390	1410	1430	1440	1140	1110	1050	950	820	690	540	380	300	210
1000	1380	1410	1430	1450	1470	1320	1280	1200	1080	940	780	620	460	370	350
900	1420	1430	1450	1480	1500	1550	1510	1430	1300	1200	980	850	750	550	500
800	1430	1450	1460	1500	1550	1600	1550	1600	1600	1600	1550	1500	1500	1550	1550
700	950	1190	1370	1500	1200	1100	1550	1600	1550	1380	1070	900	1050	1150	1200
600	910	1090	1270	1500	1200	1100	1350	1450	1200	1150	1010	880	1000	1050	1100
500	880	1060	1230	1390	1500	1500	1400	900	1100	1060	950	870	900	936	950
400	830	980	1180	1320	1450	1420	400	1300	700	900	850	810	380	780	750
300	740	880	1080	1130	1250	1280	1230	1040	900	500	700	780	750	650	550
200	650	760	880	970	1020	1050	1020	830	800	700	300	500	550	480	350
100	510	620	730	800	850	870	850	780	720	650	500	200	300	350	320
0	370	470	550	600	670	690	670	620	580	450	400	300	100	150	250
y/x	0	100	200	300	400	500	600	700	800	900	1000	1100	1200	1300	1400

解 原始数据给出的100×100网格节点上的高程数据，为了提高计算精度，我们利用双三次样条插值，得到给定区域上10×10网格节点上的高程。利用分点$x_i=10i(i=0,1,\cdots,140)$把$0\le x\le1400$剖分成 140 个小区间，利用分点$y_j=10j(j=0,1,\cdots,120)$把$0\le y\le1200$剖分成 120 个小区间，把平面区域$0\le x\le1400$，$0\le y\le1200$剖分成$140\times120$个小矩形，对应地把所计算的三维曲面剖分成$140\times120$个小曲面进行计算，每个小曲面的面积用对应的三维空间中 4 个点所构成的两个小三角形面积的和作为近似值。

计算三角形面积时，使用海伦公式，即设ΔABC的边长分别为a,b,c，$p=(a+b+c)/2$，则ΔABC的面积$s=\sqrt{p(p-a)(p-b)(p-c)}$。

利用 Python 求得的地表面积的近似值为$4.7827\times10^6\mathrm{m}^2$，所画的等高线图和三维表面图如图 7.2 所示。

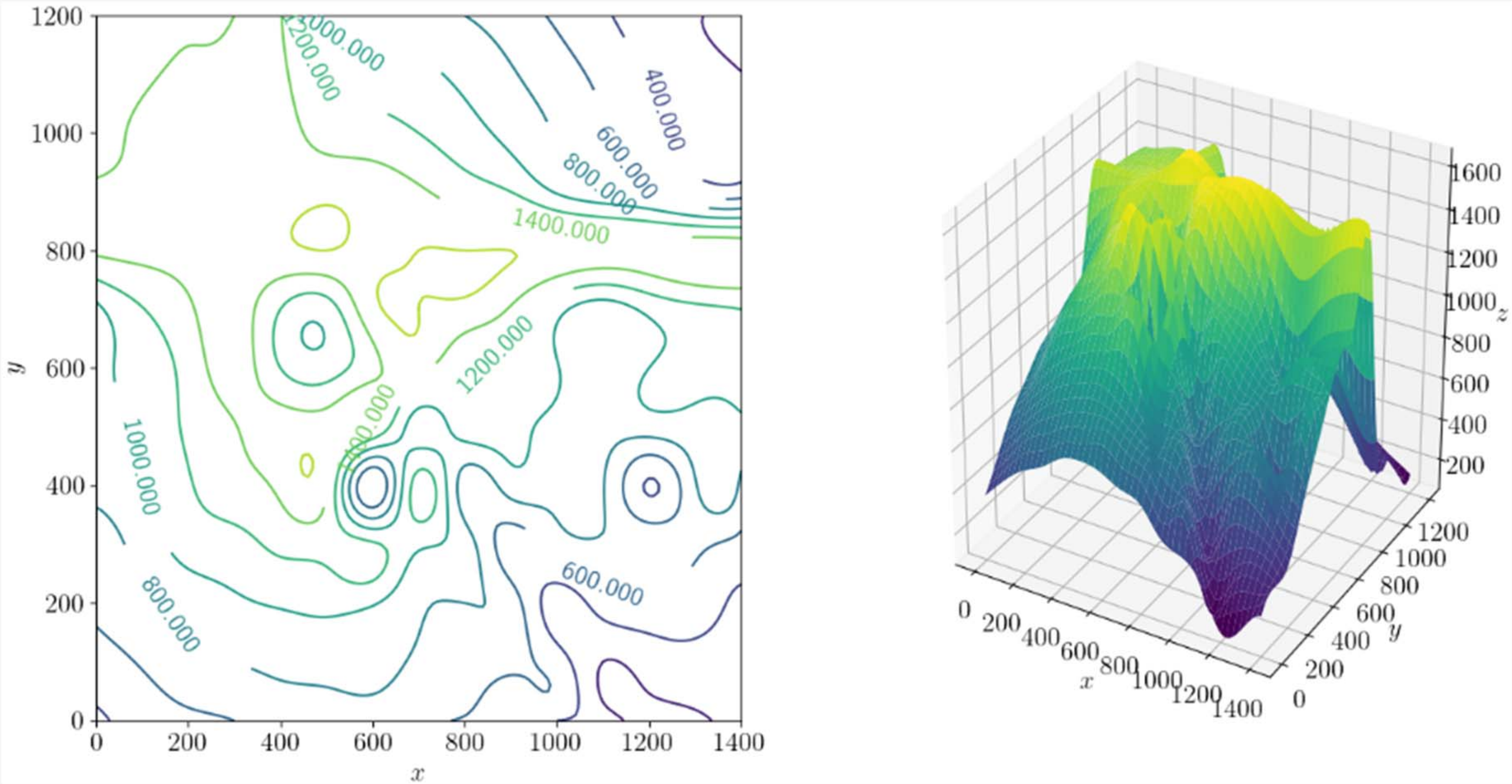

图 7.2　等高线图和三维表面图

```
#程序文件名 Pex7_5.py
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from numpy.linalg import norm
from scipy.interpolate import interp2d
z=pd.read_excel("Pdata7_5.xlsx",usecols=np.arange(1,16)) #加载高程数据
x=np.arange(0,1500,100)
y=np.arange(1200,-100,-100)
f=interp2d(x, y, z, 'cubic')
xn=np.linspace(0,1400,141)
yn=np.linspace(0,1200,121); zn=f(xn, yn)
```

```
m=len(xn); n=len(yn); s=0;
for i in np.arange(m-1):
    for j in np.arange(n-1):
        p1=np.array([xn[i],yn[j],zn[j,i]])
        p2=np.array([xn[i+1],yn[j],zn[j,i+1]])
        p3=np.array([xn[i+1],yn[j+1],zn[j+1,i+1]])
        p4=np.array([xn[i],yn[j+1],zn[j+1,i]])
        p12=norm(p1-p2); p23=norm(p3-p2); p13=norm(p3-p1);
        p14=norm(p4-p1); p34=norm(p4-p3);
        L1=(p12+p23+p13)/2;s1=np.sqrt(L1*(L1-p12)*(L1-p23)*(L1-p13));
        L2=(p13+p14+p34)/2; s2=np.sqrt(L2*(L2-p13)*(L2-p14)*(L2-p34));
        s=s+s1+s2;
```

```
print("区域的面积为： ", s)
plt.rc('font',size=16); plt.rc('text',usetex=True)
plt.subplot(121); contr=plt.contour(xn,yn,zn); plt.clabel(contr)
plt.xlabel('$x$'); plt.ylabel('$y$',rotation=90)
ax=plt.subplot(122,projection='3d');
X,Y=np.meshgrid(xn,yn)
ax.plot_surface(X, Y, zn,cmap='viridis')
ax.set_xlabel('$x$'); ax.set_ylabel('$y$'); ax.set_zlabel('$z$')
plt.savefig('figure7_5.png',dpi=500); plt.show()
```

3.二维散乱点插值

例 7.6　在某海域测得一些点（x,y）处的水深z由表 7.4 给出，画出海底区域的地形和等高线图。

表 7.4　海底水深数据

x	129	140	103.5	88	185.5	195	105	157.5	107.5	77	81	162	162	117.5
y	7.5	141.5	23	147	22.5	137.5	85.5	-6.5	-81	3	56.5	-66.5	84	-33.5
z	4	8	6	8	6	8	8	9	9	8	8	9	4	9

```
#程序文件名 Pex7_6.py
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import griddata
x=np.array([129,140,103.5,88,185.5,195,105,157.5,107.5,77,81,162,162,117.5])
y=np.array([7.5,141.5,23,147,22.5,137.5,85.5,-6.5,-81,3,56.5,-66.5,84,-33.5])
z=-np.array([4,8,6,8,6,8,8,9,9,8,8,9,4,9])
xy=np.vstack([x,y]).T
xn=np.linspace(x.min(), x.max(), 100)
yn=np.linspace(y.min(), y.max(), 100)
```

```
xng, yng = np.meshgrid(xn,yn)   #构造网格节点
zn=griddata(xy, z, (xng, yng), method='nearest')   #最近邻点插值
plt.rc('font',size=16); plt.rc('text',usetex=True)
ax=plt.subplot(121,projection='3d');
ax.plot_surface(xng, yng, zn,cmap='viridis')
ax.set_xlabel('$x$'); ax.set_ylabel('$y$'); ax.set_zlabel('$z$')
plt.subplot(122); c=plt.contour(xn,yn,zn,8); plt.clabel(c)
plt.savefig('figure7_6.png',dpi=500); plt.show()
```

输出结果如图 7.3 所示。

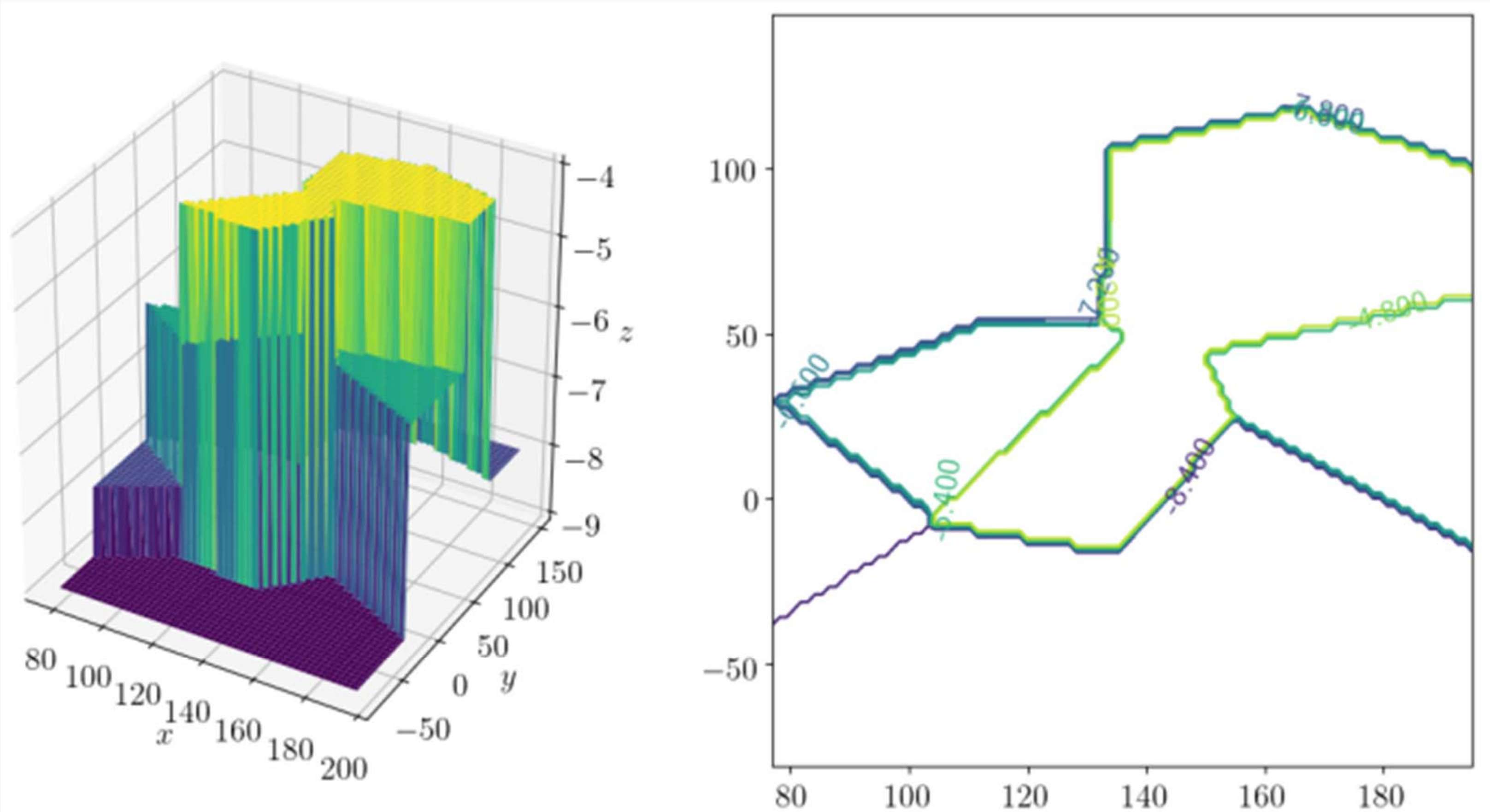

图 7.3　海底地形图及等高线图

目录 CONTENTS

7.2.1 最小二乘拟合

已知一组二维数据 $(x_i, y_i)(i=1,2,\cdots,n)$，$x_i$互不相同，求一个函数（曲线）$y=f(x)$，使$f(x)$在某种准则下与所有数据点最为接近，即曲线拟合得最好，称函数$f(x)$为拟合函数或拟合曲线。记

$$\delta_i = f(x_i) - y_i, \quad i=1,2,\cdots,n,$$

则称δ_i为拟合函数$f(x)$在x_i点处的偏差（或残差）。为使$f(x)$在整体上尽可能与给定数据最为接近，可以采用“偏差的平方和最小”作为判定准则，即求函数$f(x)$，使得

$$J = \sum_{i=1}^{n} (f(x_i) - y_i)^2 \quad (7.5)$$

达到最小值。这一原则称为最小二乘原则，根据最小二乘原则求拟合函数 $f(x)$的方法称为最小二乘法。

通常拟合函数除了含有自变量x外，还含有待定参数$a_1,a_2,\cdots,a_m$，即

$$f(x)=f(x,a_1,a_2,\cdots,a_m). \tag{7.6}$$

按照$f(x)$关于参数$a_1,a_2,\cdots,a_m$是否线性，最小二乘法分为线性最小二乘法和非线性最小二乘法两类。

1.线性最小二乘法

设有n个数据点$(x_i, y_i)(i=1,2,\cdots,n)$，$x_i$互异，以及线性无关的函数系$\{\varphi_k(x) \mid k=1,2,\cdots,m\}$，其中 n⩾m。如果拟合函数是线性组合的形式

$$f(x)=\sum_{k=1}^{m} a_k \varphi_k(x) \tag{7.7}$$

则$f(x)=f(x,a_1,a_2,\cdots,a_m)$就是关于参数$a_1,a_2,\cdots,a_m$的线性函数。

例如拟合函数

$$f(x)=a_m x^{m-1}+a_{m-1}x^{m-2}+\cdots+a_2 x+a_1\quad,\quad f(x)=\sum_{k=1}^{m} a_k\cos(kx).$$

将式（7.7）代入式（7.5），则目标函数$J=J(a_1,a_2,\cdots,a_k)$是关于参数$a_1,a_2,\cdots,a_m$的多元函数。由

$$\frac{\partial J}{\partial a_k}=0,\quad k=1,2,\cdots,m,$$

亦即

$$\sum_{i=1}^{n}[(f(x_i)-y_i)\varphi_k(x_i)]=0,\quad k=1,2,\cdots,m.$$

可得

$$\sum_{j=1}^{m}\left[\sum_{i=1}^{n}\varphi_j(x_i)\varphi_k(x_i)\right]a_j=\sum_{i=1}^{n}y_i\varphi_k(x_i),\quad k=1,2,\cdots,m. \tag{7.8}$$

于是式 (7.8) 形成了一个关于$a_1, a_2, \cdots, a_m$的线性方程组，称为正规方程组。

记

$$R = \begin{bmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_m(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_m(x_2) \\ \vdots & \vdots & & \vdots \\ \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_m(x_n) \end{bmatrix}, \quad A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

则正规方程组（7.8）可表示为

$$R^T RA = R^T Y. \tag{7.9}$$

因为函数系$\{\varphi_k(x)\,|\,k=1,2,\cdots,m\}$线性无关,并且n ≥ m，所以矩阵$R$是列满秩，故矩$R^T R$可逆，正规方程组（7.9）有唯一解，且解表示为

$$A=(R^T R)^{-1} R^T Y \qquad (7.10)$$

为所求的拟合函数的系数，就可得到最小二乘拟合函数$f(x)$。

2.非线性最小二乘拟合

如果拟合函数 $f(x)$不能写成某个线性无关函数系$\{\varphi_k(x)\,|\,k=1,2,\cdots,m\}$的线性组合形式，则 $f(x)=f(x,a_1,a_2,\cdots,a_m)$关于参数$a_1,a_2,\cdots,a_m$是非线性函数。例如

$$f(x)=\frac{x}{a_1x+a_2} \text{或者} f(x)=a_1+a_2e^{-a_3x}+a_4e^{-a_5x}.$$

将 $f(x)$代入式（7.5）中，则形成一个非线性函数的极小化问题。为得到最小二乘拟合函数 $f(x)$的具体表达式，可用非线性优化方法求解出参数$a_1,a_2,\cdots,a_m$。

有些非线性拟合问题可以化为线性拟合问题. 例如，对非线性拟合函数

$$f(x) = \frac{x}{a_1 x + a_2},$$

令$\varphi_1(x) = 1, \varphi_2(x) = 1/x$, 则

$$g(x) = \frac{1}{f(x)} = a_1 + \frac{a_2}{x} = a_1\varphi_1(x) + a_2\varphi_2(x)$$

是线性拟合函数. 一般的非线性最小二乘拟合问题可用非线性优化方法求解.

3.拟合函数的选择

数据拟合时，第一步就是选取恰当的拟合函数。确定拟合函数一般有两种方法:

（1）如果能够根据问题的背景通过机理分析得到变量之间的函数系，那么可以确定拟合函数。

（2）利用给出的数据做散点图，从直观上判断应选用什么样的拟合函数。

当确定拟合函数后，利用拟合方法求出拟合函数中的参数，就得到最终结果。

根据散点图确定拟合函数类型有以下几种形式:

(1) 如果数据分布接近于直线，则可以选用线性函数 $f(x)=a_1x+a_2$ 拟合;

(2) 如果数据分布接近于抛物线，则拟合函数可以用二次多项式

$$f(x)=a_1x^2+a_2x+a_3.$$

(3) 如果数据分布特点是开始上升较快随后逐渐变缓，则选如下的双曲线型函数或指数型函数作为拟合函数

$$f(x)=\frac{x}{a_1x+a_2} \text{或} f(x)=a_1e^{-\frac{a_2}{x}}.$$

（4）如果数据分布特点是下降的并且开始下降较快随后逐渐变缓，则可用

$$f(x)=\frac{1}{a_1x+a_2}，\quad f(x)=\frac{1}{a_1x^2+a_2}\text{或}f(x)=a_1e^{-a_2x}$$

等作为拟合函数。

（5）利用数据分布特点，常用拟合函数还有对数函数 $y=a_1+a_2\ln x$，S 形曲线函数 $y=\dfrac{1}{a+be^{-x}}$ 等。

7.2.2 数据拟合的Python实现

Python 的多个模块中，有很多函数或方法可以拟合未知参数。例如NumPy 库中的多项式拟合函数 polyfit；scipy.optimize 模块中的函数leastsq，curve_fit 都可以拟合函数。下面介绍 polyfit 和 curve_fit 的使用方法。

1.polyfit 的用法

例 7.7 对表 7.5 的数据进行二次多项式拟合。并求当 $x = 0.25, 0.35$ 时 y 的预测值。

表 7.5 拟合数据

x_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y_i	-0.447	1.978	3.28	6.16	7.08	7.34	7.66	9.56	9.48	9.30	11.2

解：#程序文件名 Pex7_7.py

```
from numpy import polyfit, polyval, array, arange
from matplotlib.pyplot import plot,show,rc
x0=arange(0, 1.1, 0.1)
y0=array([-0.447, 1.978, 3.28, 6.16, 7.08, 7.34, 7.66, 9.56, 9.48, 9.30, 11.2])
p=polyfit(x0, y0, 2) #拟合二次多项式
print("拟合二次多项式的从高次幂到低次幂系数分别为:",p)
yhat=polyval(p,[0.25, 0.35]); print("预测值分别为：  ", yhat)
rc('font',size=16)
plot(x0, y0, '*', x0, polyval(p, x0), '-'); show()
```

拟合的二次多项式为 $y = -9.8108x^2 + 20.1293x - 0.0317$； $x = 0.25, 0.35$时，y的预测值分别为 4.3875， 5.8118。

2.curve_fit

curve_fit 的调用格式为：

$$\text{popt, pcov = curve\_fit(func, xdata, ydata)}$$

其中 func 是拟合的函数，xdata 是自变量的观测值，ydata 是函数的观测值，返回值 popt 是拟合的参数，pcov 是参数的协方差矩阵。

例 7.8（续例 7.7） 用 curve_fit 函数拟合二次多项式，并求预测值。

```
#程序文件名 Pex7_8.py
import numpy as np
from scipy.optimize import curve_fit
y=lambda x, a, b, c: a*x**2+b*x+c
x0=np.arange(0, 1.1, 0.1)
y0=np.array([-0.447, 1.978, 3.28, 6.16, 7.08, 7.34, 7.66, 9.56, 9.48, 9.30, 11.2])
popt, pcov=curve_fit(y, x0, y0)
print("拟合的参数值为：", popt)
print("预测值分别为：", y(np.array([0.25, 0.35]), *popt))
```

运行结果如下：

拟合的参数值为：[-9.81083901 20.12929291 -0.03167108]

预测值分别为：[4.38747471 5.81175366]

例 7.9　用表 7.6 的数据拟合函数 $z = ae^{bx} + cy^2$。

表 7.6　x_1, x_2, y 的观测值

x	6	2	6	7	4	2	5	9
y	4	9	5	3	8	5	8	2
z	5	2	1	9	7	4	3	3

```
#程序文件名 Pex7_9.py
import numpy as np
from scipy.optimize import curve_fit
x0=np.array([6, 2, 6, 7, 4, 2, 5, 9])
y0=np.array([4, 9, 5, 3, 8, 5, 8, 2])
z0=np.array([5, 2, 1, 9, 7, 4, 3, 3])
xy0=np.vstack((x0, y0))
def Pfun(t, a, b, c):
    return a*np.exp(b*t[0])+c*t[1]**2
popt, pcov=curve_fit(Pfun, xy0, z0)
```

print("a，b，c 的拟合值为：", popt)

求得($a = 5.0891$，$b = -0.0026$，$c = -0.0215$。) a=1.80474379, b=-8.21727391, c=0.06509113

例 7.10 利用模拟数据拟合曲面$z=e^{-\frac{(x-\mu_1)^2+(y-\mu_2)^2}{2\sigma^2}}$，并画出拟合曲面的图形。

我们利用函数$z=e^{-\frac{(x-\mu_1)^2+(y-\mu_2)^2}{2\sigma^2}}$，其中$\mu_1=1$，$\mu_2=2$，$\sigma=3$，生成加噪声的模拟数据，利用模拟数据拟合参数$\mu_1,\mu_2,\sigma$，最后画出拟合曲面的图形。

```
#程序文件 Pex7_10.py
from mpl_toolkits import mplot3d
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
m=200; n=300
x=np.linspace(-6, 6, m); y=np.linspace(-8, 8, n);
x2, y2 = np.meshgrid(x, y)
x3=np.reshape(x2,(1,-1)); y3=np.reshape(y2, (1,-1))
xy=np.vstack((x3,y3))
```

```
def Pfun(t, m1, m2, s):
    return np.exp(-((t[0]-m1)**2+(t[1]-m2)**2)/(2*s**2))
z=Pfun(xy, 1, 2, 3); zr=z+0.2*np.random.normal(size=z.shape) #噪声数据
popt, pcov=curve_fit(Pfun, xy, zr)    #拟合参数
print("三个参数的拟合值分别为：",popt)
zn=Pfun(xy, *popt)   #计算拟合函数的值
zn2=np.reshape(zn, x2.shape)
plt.rc('font',size=16)
ax=plt.axes(projection='3d') #创建一个三维坐标轴对象
ax.plot_surface(x2, y2, zn2,cmap='gist_rainbow')
plt.savefig("figure7_10.png", dpi=500); plt.show()
```

拟合曲面的图形如图 7.4 所示。

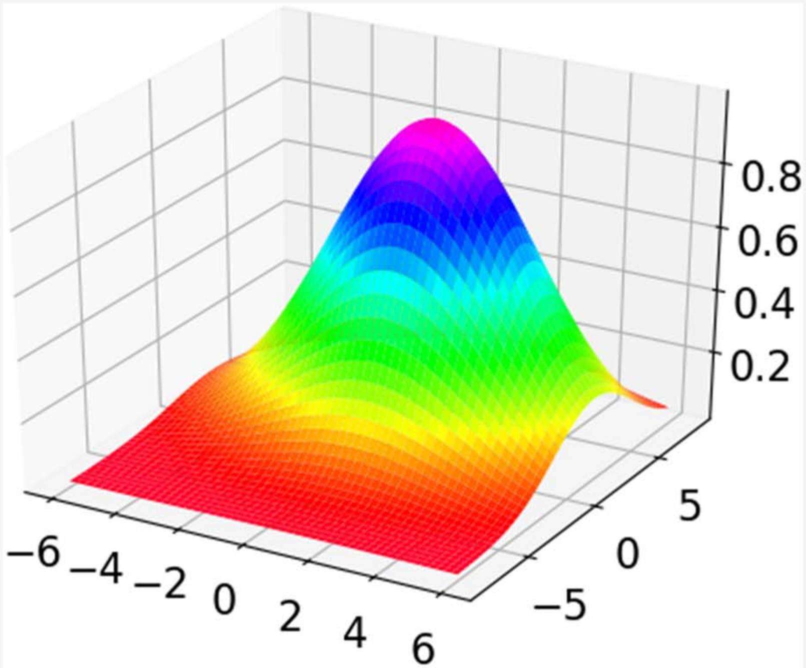

图 7.4　拟合曲面的图形