北京航空航天大學
BEIHANG UNIVERSITY




# 大学计算机基础

## （理科类）

## 第12讲 数据处理

北京航空航天大学

# 目　录

# 本讲重点和难点

**重点**

- 了解Python常用的科学计算工具**NumPy、Scipy**和**Pandas**
- 掌握数据拟合中**最小二乘拟合**方法
- 掌握**一维插值、二维插值**的方法

**难点**

- 数据拟合：如何根据数据分布规律，确定出**合适**的**拟合函数**模型？

北京航空航天大學
BEIHANG UNIVERSITY



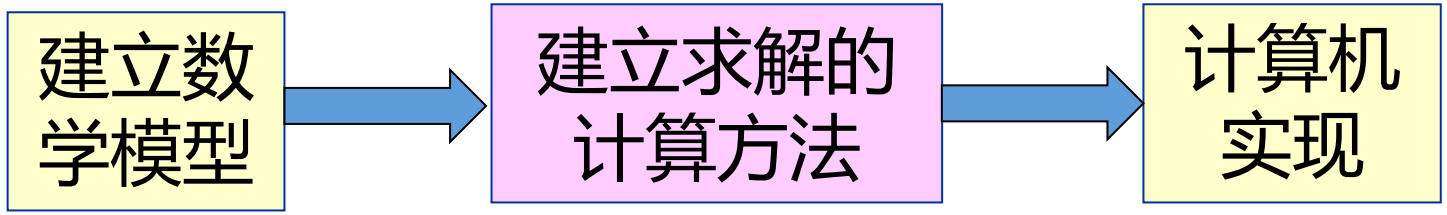

# 12.1 Python科学计算工具

北京航空航天大学

# 什么是科学计算？

- **科学计算**是**为解决科学和工程中的数学问题，利用计算机求数学方程式的数值解**
- 常见的数据处理算法包括**数据拟合、参数估计、插值**等

- **科学计算过程**



# Python常用科学计算工具简介

- **Python用于科学计算、数据分析与可视化的第三方库**
  - **NumPy**：**科学计算基础库。**创建N维数组、线性代数计算、三角函数计算、多项式拟合、傅立叶变换、生成随机数等
  - **Scipy**：**数值计算库。**线性代数计算、拟合与优化、插值、数值积分、稀疏矩阵、图像处理、统计等
  - **Pandas**：**数据分析库。**数据导入、整理、处理、分析等
  - **Matplotlib**：**绘图库。**绘制二维图形和图表

# 一、NumPy：科学计算基础库

## NumPy的主要功能

- **创建一维、二维、N维数组，支持N维数组运算**
  - ✓ **数组的优势**：不必编写循环即可对数据执行**批量运算**
  - ✓ 大小相等的数组之间的任何算术运算都会将运算应用到**元素级**
- **内置许多ufunc（通用）函数，可以对一个数组或两个数组进行算术运算、三角运算**
- 处理**大型矩阵**，成熟的广播函数库，矢量运算
- 线性代数，傅里叶变换、随机数生成
- ……

# 数组

- **数组**是用方括号括起来的一系列元素，各元素之间用一个空格分隔。例： [5 4 3.2 8 1]
- 可以采用**数组名[<索引>]**的方式访问数组中的元素，或者采用**切片**的方式访问连续几个元素（不包括上限）
- NumPy的**算术运算函数**
  - **加法**：add(x1,x2[,y])　　**减法**： subtract(x1,x2[,y])
  - **乘法**： multiply(x1,x2[,y])　　**除法**： devide(x1,x2[,y])
  - **幂运算**： power(x1,x2[,y])

- **函数ones(n)和zeros(n) 创建长度为n的全1或全0数组**

```
>>> np.ones(6)
array([ 1.,  1.,  1.,  1.,  1.,  1.])
```

```
>>> np.zeros(6)
array([ 0.,  0.,  0.,  0.,  0.,  0.])
```

# NumPy的常用函数

| 序号 | 函数名称 | 功能 |
| --- | --- | --- |
| 1 | array([x,y,z], dtype=int) | 将输入数据转换为**一维或多维数组**。可以通过dtype参数在创建时指定元素类型。默认直接复制输入数据 |
| 2 | arange(<开始值>,<终值>,<步长>) | 通过指定开始值、终值和步长来**创建一维数组**，注意数组不包括终值 |
| 3 | linspace(<开始值>,<终值>,<元素个数>) | 通过指定开始值、终值和元素个数来**创建一维数组**，可以通过endpoint关键字指定是否包括终值，缺省设置是包括终值 |
| 4 | sin(x) | 对数组x中的每个元素进行**正弦计算**，返回一个同样大小的新数组 |

# NumPy的常用函数（续1）

| 序号 | 函数名称 | 功能 |
| --- | --- | --- |
| 4 | cos(x) | 对数组x中的每个元素进行**余弦计算**，返回一个同样大小的新数组 |
| 5 | add(x1,x2[,y]) | 两个数组x1与x2中对应元素**相加**，可选参数y指定计算结果所要写入的数组 |
| 6 | polyfit(x,y,n) | **多项式拟合**函数。x,y为要拟合的数据，n为拟合多项式的次数，返回值为拟合多项式系数 |

- **三角运算函数还有tan、arcsin、arccos、arctan等**

# NumPy的常用函数（续2）

| 序号 | 函数名称 | 功能 |
| --- | --- | --- |
| 7 | poly1d(c_or_r, r=False, variable=None) | **一维多项式类**。参数1：为一个数组，若没有参数2，则生成一个多项式。若参数2为True，则表示把数组中的值作为根，然后反推多项式。参数3：variable='z'表示改变未知数的字母。<br>构建多项式p后，p(x) 计算多项式在某个x处的值；<br>p.r求多项式的根；p.c显示多项式的系数；<br>p.order显示多项式的阶数；p[1]返回第一项的系数；多项式支持实数的四则运算 |
| 8 | polyval(p,x) | **多项式求值函数**。计算多项式在x处的值，其中p为拟合多项式系数 |

# 1、array函数的用法

- **array函数**将输入数据转换为**一维**数组或**多维**数组
- 它接受一切**序列型**的对象（**列表，元组，数组**），然后产生一个新的含有传入数据的NumPy数组
- **嵌套序列**（**等长**）将会被转换为一个**多维**数组

**格式** array(<序列型的对象>, dtype=<指定元素类型>)

# 【例12.1 】 array函数创建一维数组

- 【例12.1】利用array函数将一个列表转换为一维数组。

- 可以通过dtype参数在创建时指定元素类型。

```
>>> import numpy as np
>>> lis1=[5, 4, 3.2, 8, 1]
>>>arr1=np.array(lis1, dtype=float)
>>>arr1
array([ 5. ,  4. ,  3.2,  8. ,  1. ])
```

浮点数

# 【例12.2 】 array函数创建多维数组

■ **【例12.2】**利用array函数将一个嵌套列表转换为多维数组。

```
>>> import numpy as np
>>> lis2=[[1, 2, 3, 4], [5, 6, 7, 8]]
>>>arr2=np.array(lis2)
>>>arr2
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

# dtype属性和shape属性

- 数组的**元素类型**可以通过**dtype**属性获得
- 数组的**大小**可以通过其**shape**属性获得

```
>>> import numpy as np
>>> lis2=[[1, 2, 3, 4], [5, 6, 7, 8]]
>>>arr2=np.array(lis2)
>>>arr2.dtype      #获取数组的元素类型
dtype('int32')
>>>arr2.shape      #获取数组的大小
(2, 4)
```

# array函数的用法

```
>>> a = [1, 2, 3, 4]
>>> b = (5, 6, 7, 8)
>>> c = np.array([a,b])
>>> c
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

```
>>> a = (1, 2, 3, 4)
>>> b = (5, 6, 7, 8)
>>> d = np.array((a,b))
>>> d
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

- 无论a、b是列表还是元组，“np.array([a,b]) ”都会将[a,b] 转换为**二维**数组

- **array函数接受一切序列型的对象（列表，元组，数组）**

# arange函数和linspace函数

- **创建数组的一般方法**
  - 先创建一个**序列**（列表或元组）
  - 再利用**array函数**将其转换为数组
- **如何快速创建元素间隔一定步长的一维数组?**
  - 创建（一维）数组的函数：**arange**函数、**linspace**函数

**格式** `arange(<开始值>,<终值>,<步长>)`

**格式** `linspace(<开始值>,<终值>,<元素个数>)`

# 二、Scipy：数值计算库

- Scipy提供了**更多的数学工具**，包括矩阵运算、线性方程组求解、积分、优化、**插值**、**拟合**、信号处理、图像处理、统计等（常用模块共**16**个）
- Scipy依赖于NumPy
- Numpy与Scipy有机结合，完全可以替代Matlab的计算功能（包括其插件工具箱）

# interpolate模块

- **interpolate：插值**模块，提供了许多对数据进行插值运算的函数
  - 一维插值函数**interp1d()**
  - 二维插值函数**interp2d()**
  - 多维插值函数**griddata()**
- 将在“12.3 函数插值”中详细介绍**interp1d()**和**interp2d()**

# optimize模块

- **optimize**：**最优化函数库**，提供了进行最优化和求等式的根的函数
  - 求函数最小值：标量**minimize_scalar()**或多维**minimize()**）
  - 曲线拟合：**leastsq()**，实现**最小二乘拟合**算法
  - 非线性方程组求解：**fsolve()**函数
- 将在“12.2 数据拟合”中详细介绍**leastsq()**

# 三、Pandas：数据分析库

- Pandas是基于NumPy的**数据分析**库
- 它提供了大量**高级数据结构**和能够高效便捷地操作大型数据集的工具

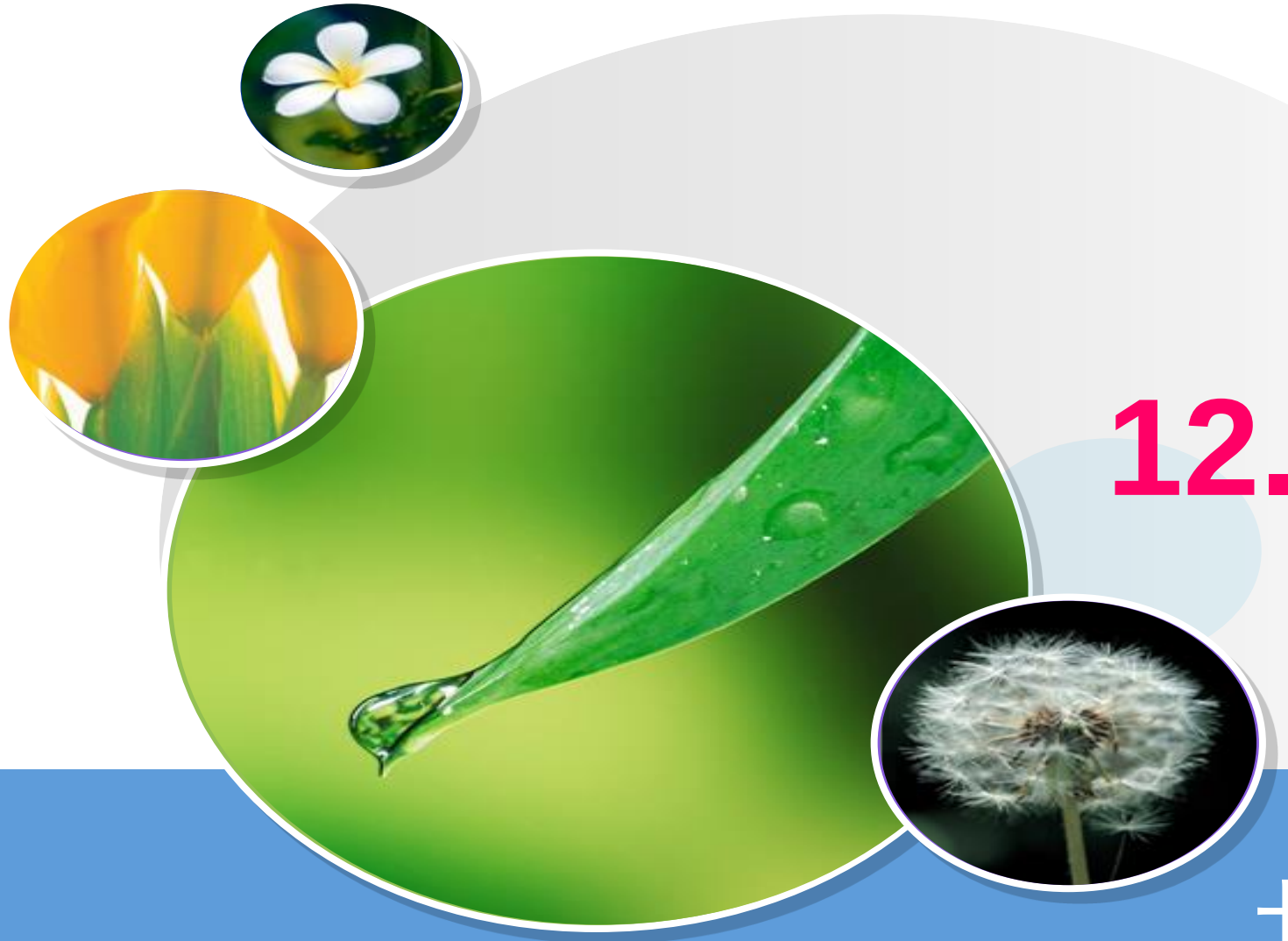
（1）**Series**：带标签的一维数组，由一组数据（各种NumPy数据类型）和与之相关的数据标签（索引）组成；

（2）**DataFrame**：带标签且大小可变的二维表格结构；

（3）**Panel**：带标签且大小可变的三维数组

- Pandas提供了大量的**函数**用于**生成、访问、修改、保存**不同类型的数据，处理缺失值、重复值、异常值，并能够结合Matplotlib绘图库进行数据可视化

**在第13讲讲授数据分析时介绍Pandas的应用**



北京航空航天大學
BEIHANG UNIVERSITY

# 12.2 数据拟合

北京航空航天大学

# 12.2 数据拟合

- 12.2.1 插值问题与拟合问题
- 12.2.2 数据拟合的分类和主要方法
- 12.2.3 线性最小二乘拟合
- 12.2.4 非线性最小二乘拟合

# 12.2.1 插值问题与拟合问题

- 在**生产（或工程）实践**和**科学实验**过程中，通常需要通过研究某些变量之间的**函数关系**来认识事物的**内在规律**和**本质属性**
- 这些变量之间的未知函数关系常常隐含在从实验观察、测量得到的一组数据之中
  - 无法写出解析表达式
  - 解析表达式过于复杂
- **能否根据一组实验观测数据**$(x_i,y_i)$，$i=1,2,\ldots n$**找到变量之间相对准确的函数关系**$y=g(x)$**?**

# 寻找近似函数$y=f(x)$的两类方法

- 希望找到比较简单的近似函数y=f(x)，使函数在观测点的值**等于**或者**接近**观测值，用函数y=f(x)取代原始的函数关系y = g(x)
- **寻找近似函数$y=f(x)$的两类方法**
  - （1）当测量数据的数据量较小并且数据值是准确的，或者基本没有误差时，一般用**插值**的方法
  - （2）当通过测量得到的数据比较多，或者测量值与真实值之间有一定误差时，一般用**拟合**的方法

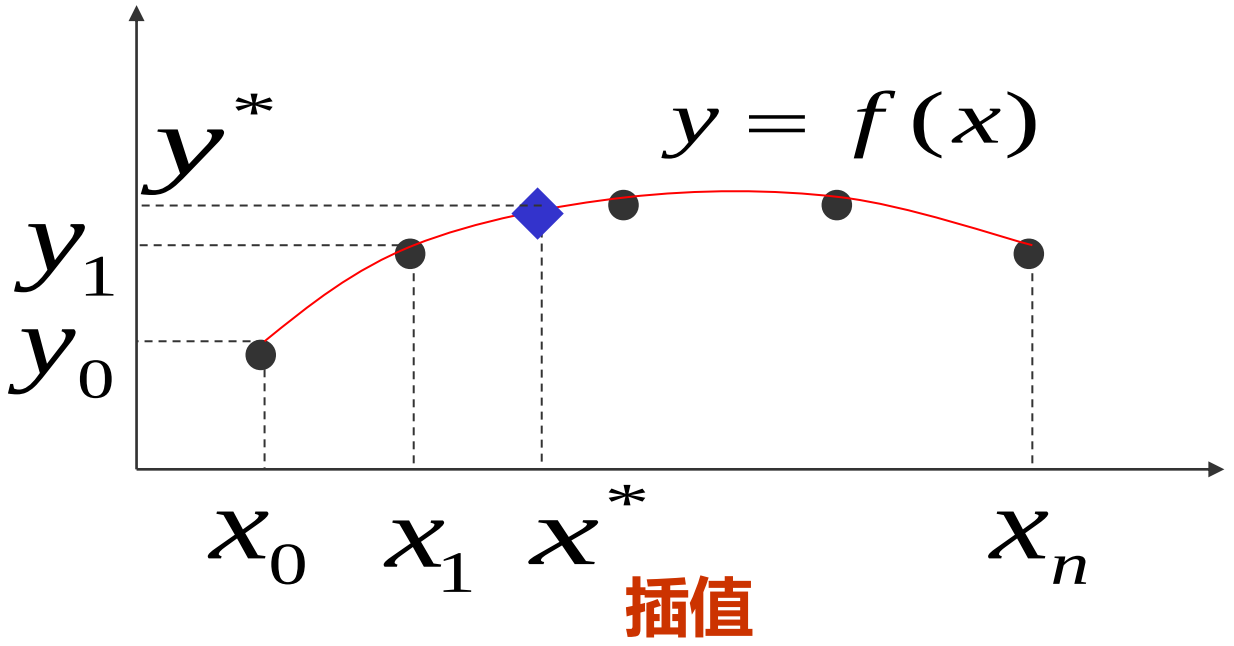
# 1. 插值问题

- **【引例12.1 】**已知在一天24小时内，从零点开始每间隔2小时测得的环境温度数据分别为：12，9，9，10，18 ，24，28，27，25，20，18，15，13。试推测中午1点的温度，并作出24小时内的温度变化曲线图。

- 问题归结为：“**已知函数在某区间（域）内若干点处的值，求函数在该区间（域）内其它点处的值**”
- 此类问题称为**插值问题**
- 插值问题适宜用插值方法解决

# 什么是插值？

- **用一个近似的函数关系式y = f（x）来刻画一组实验观测数据中自变量x与因变量y之间的关系，要求这个近似函数曲线通过已知的所有数据点，**这种方法称为**插值**（Interpolation）



# 2. 拟合问题

■ 【引例12.2】已知某合金成分x与膨胀系数y之间的关系有如下表所示的实验数据。试求膨胀系数y与成分x的近似函数关系$y=f(x)$。并推测合金成分为45时的膨胀系数。

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| x | 37 | 38 | 39 | 40 | 41 | 42 | 43 |
| y | 3.40 | 3.00 | 2.10 | 1.53 | 1.80 | 1.90 | 2.90 |

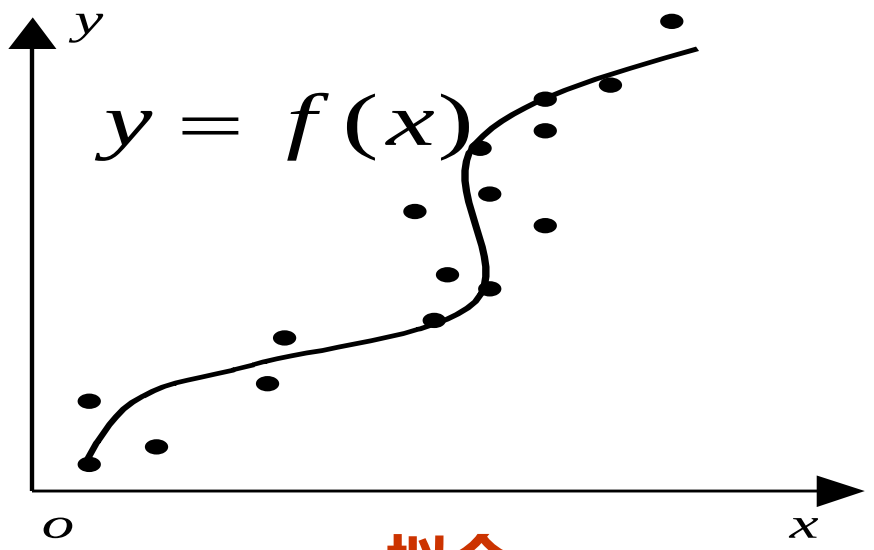
■ 问题归结为：**已知函数在某区间（域）内若干点处的值，求函数表达式，以及函数在该区间（域）外其它点处的值**

# 解决思路

- **不宜采用插值方法**
  - 因为45已超出所给数据范围[37，43]，如果用插值函数**外推**插值区间外的数据，将会产生**较大误差**
- **解决方法**：根据[37,43]间的膨胀系数数据，求出膨胀系数与合金成分之间的近似函数关系**f(x)**，然后由f(x)推断x =45时的温度

# 什么是拟合？

- **用一个近似的函数关系式y = f（x）来刻画一组实验观测数据中自变量x与因变量y之间的关系，不要求近似函数曲线通过所有数据点，**这种方法称为**拟合**（Fitting）
- 得到比原函数**更简单**、**更适用**的近似函数
- 反映原函数整体的变化趋势




拟合

# 12.2.2 数据拟合的分类和主要方法

1. 数据拟合的分类

2. 曲线拟合常用的拟合函数类型

3. 最小二乘拟合

# 1. 数据拟合的分类

- **数据拟合**：**选择适当的曲线（或曲面）类型来拟合实验观测数据，并用拟合函数近似表示因变量与自变量之间的函数关系**
- 根据自变量的个数，数据拟合分为
  - **曲线拟合**（只有**一个**自变量）
  - **曲面拟合**（有**两个**自变量）

- 数据拟合必须确定**函数表达式**，然后确定其中的**参数**

# 2. 曲线拟合常用的拟合函数类型

- **直线** $y = a_1 x + a_0$
- **多项式** $y = a_n x^n + \cdots + a_1 x + a_0$
- **双曲线（一支）** $y = \frac{a_1}{x} + a_2$
- **指数曲线** $y = a_1 e^{a_2 x}$
- **三角函数**

sin(x)， cos(x)， sin(2x)， cos(2x)， ……，
sin(mx)， cos(mx)

# 用什么样的曲线拟合已知数据?

- **用什么样的曲线拟合已知数据?**
  - **方法一**：画散点图→观察实验数据的规律→确定曲线的函数类型（**有时需要几个函数组合**） →根据**最小二乘原则**，确定拟合函数中的**未知参数**
  - **方法二**：理论分析→初步选定曲线函数，建立数学模型

# 3. 最小二乘拟合

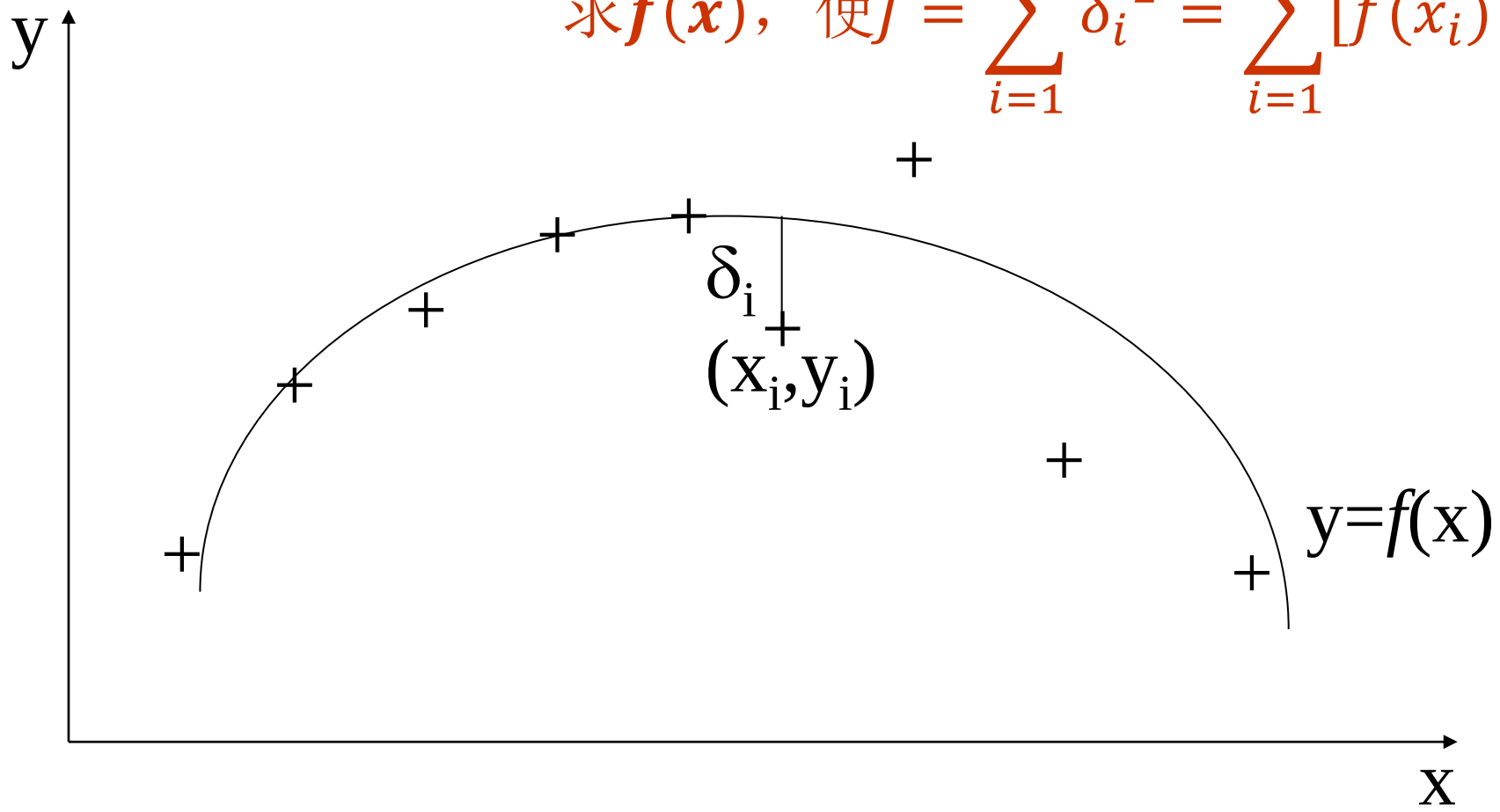
- **拟合**：寻求一个**简单**、**易算**的近似函数$y = f(x)$，使$f(x)$在**某种准则**下与已知的n个离散数据点（$x_i$ $y_i$）（$i$=1,2,…,$n$）最接近
- 拟合的**关键**就是选择什么样的**准则**来评价拟合精度，使得偏差$\delta_i = f(x_i) - y_i$总体上尽可能地小
- 工程中许多问题都归结为**偏差平方和**$J$达到**最小**

$$J = \sum_{i=1}^{n} {\delta_i}^2 = \sum_{i=1}^{n} [f(x_i) - y_i]^2$$

- **这一条件称为最小二乘原则（或最小二乘准则）**

# 最小二乘原则示意图

求$\boldsymbol{f(x)}$，使$J=\sum_{i=1}^{n}\delta_i{}^2=\sum_{i=1}^{n}[f(x_i)-y_i]^2$最小




$\delta_i$ 为点 $(x_i,y_i)$ 与曲线 y=f(x) 的距离

# 最小二乘拟合方法

- **最小二乘拟合：通过使各实验（或观测）数据与拟合曲线的偏差的平方和最小，来寻找实验数据的最佳函数匹配的方法**
- **最小二乘拟合方法分为两种**
  - 线性最小二乘拟合
  - 非线性最小二乘拟合
- 假设y = f(x,b)为拟合模型，其中x为自变量，b为待定参数，如果拟合模型f(x,b)关于**待定参数**b是**线性**的，则称模型为**线性**的，采用的方法为**线性最小二乘拟合方法**
- 否则模型为**非线性**的，采用的方法为**非线性最小二乘拟合方法**

# 12.2.3 线性最小二乘拟合

- **线性最小二乘拟合方法**
  - **方法一**：采用**Scipy**库的**optimize**模块中实现最小二乘拟合算法的函数**leastsq()**，求出待定系数
  - **方法二**：采用**多项式拟合，** 利用**NumPy**库的多项式拟合函数**polyfit()函数**求解多项式系数$a_1$、$a_0$

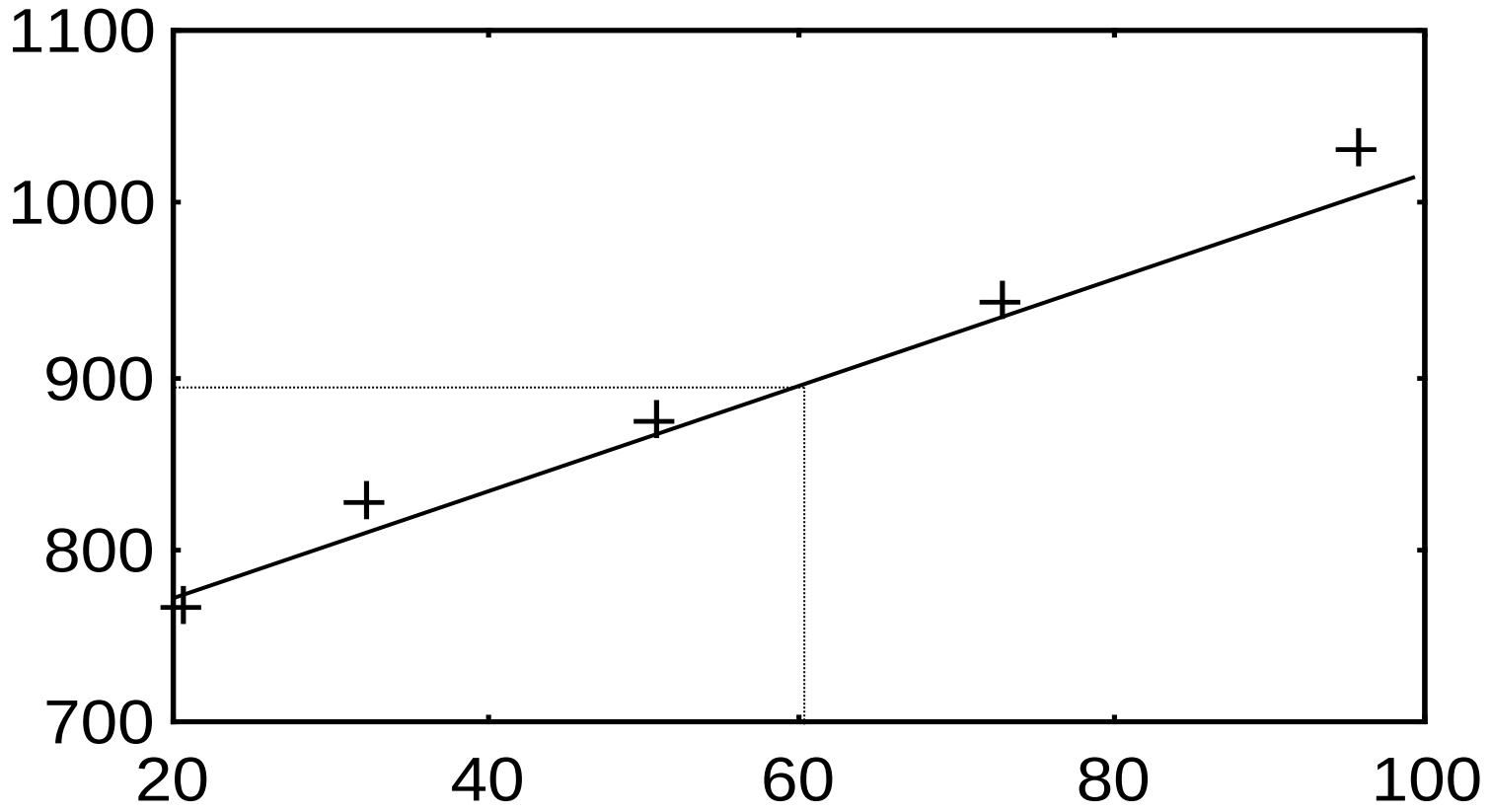
$$f(x) = a_1 x + a_0$$

# 【例12.0】热敏电阻问题

- 【例12.0 】对于某个热敏电阻，测得其在某些特定温度下的电阻值如下表所示，试根据实验数据求出**拟合函数**，并估计其在**60°C**时的电阻R。

| 温度t(°C) | 20.5 | 32.7 | 51.0 | 73.0 | 95.7 |
|---|---|---|---|---|---|
| 电阻R(Ω) | 765 | 826 | 873 | 942 | 1032 |

# 【例12.0】求解思路




**作图**，绘制离散数据点

- **观察**，预测属于什么**曲线**，确定其**模型**：**R=at+b**
- **编程**，求出待定系数a、b

# leastsq()函数

- **Scipy**库中的**optimize**模块提供了实现最小二乘拟合算法的函数**leastsq()**
  - 根据给出的一系列数据点，对给定的待拟合函数实现最小二乘拟合算法，求出**拟合参数**
  - 使用时必须先从Scipy库的optimize模块中**导入**该函数

**from scipy.optimize import leastsq**

**leastsq**(*func, x0, args=(), Dfun=None, full_output=0, col_deriv=0, ftol=1.49012e-08, xtol=1.49012e-08, gtol=0.0, maxfev=0, epsfcn=None, factor=100, diag=None*)

# leastsq()函数的用法

偏差函数

待拟合参数的初始值

实验数据

**leastsq(*func, x0, args=(), Dfun=None, full_output=0, col_deriv=0, ftol=1.49012e-08, xtol=1.49012e-08, gtol=0.0, maxfev=0, epsfcn=None, factor=100, diag=None*)**

- 尽管leastsq函数的参数很多，但大部分是可选参数
- 一般传入前三个参数**func**、**x0**和**args**即可

# leastsq()函数的主要参数

**（1）输入参数**

**func**：**偏差函数**（计算**拟合**数据与**实验**数据之间的误差），可调用

**x0**：N维数组，**待拟合参数的初始值**

**args**：需要拟合的**实验数据**（必须存储在**数组**里），用**元组**表示，如(x,y)

**……**

**（2）返回值**

**x**：N维数组，**求解的拟合参数**

**……**：其它返回值

# 【例12.0】采用leastsq()函数求解

- **【例12.0】求解方法一**
  - 拟合模型为：**R=at+b**，故采用**线性最小二乘拟合**求解
  - 假定偏差函数的函数名为residuals，拟合参数a、b的初始值为1、1，需要拟合的一组实验数据x、y分别存储在数组t、R中

    **r = leastsq(residuals, [1,1], args=(t, R))**

  - leastsq函数返回值的第一个值r[0]，即为所求解的拟合参数a，b

# 【例12.0】编程思路

- **偏差函数**计算实验数据和拟合数据之间的误差。假定实验数据的因变量用y表示，拟合数据的因变量用y*表示，则误差**error= y*-y**
- 关键是**拟合数据具体如何表示**?
  - 拟合数据是根据拟合函数计算得到的，即**R=at+b**。
  - 可以先定义一个**待拟合的函数fun(x, p)**
    - ✓ 其参数为x、p，其中x是自变量（即公式中t），p是待拟合的参数（一维数组 [a, b]）
    - ✓ fun(x, p)的返回值即为**a*x+b**

# 【例12.0】编程思路（续）

- 然后**定义偏差函数residuals(p, x, y)**
  - 参数p是待拟合的参数（一维数组形式），x和y分别是对应真实数据的x、y值（**一维数组**）
  - 返回值为偏差“**fun(x, p) – y**”

- 假定实验数据的x、y值分别存储在两个列表中，则可以使用numpy库的**array函数**将**列表**转换为**一维数组**

# 【例12.0】Python程序

**例12.0-thermistor-leastsq拟合.py**

```
import numpy as np
#scipy中的optimize模块提供了实现最小二乘拟合算法的函数leastsq
from scipy.optimize import leastsq

#（1）定义待拟合的函数。x是自变量，p是待拟合的参数（一维数组）
def fun(x, p):
    a, b = p    #数组p中的元素分别赋给a、b这两个变量
    return a*x+b
```

f=at+b

# 【例12.0】Python程序（续1）

```
#（2）定义偏差函数：计算实验数据与拟合数据之间的误差。p是待拟合的参数，x和y分别是对应的实验数据的x、y值
def residuals(p, x, y):
    return fun(x, p) – y                    #调用待拟合的函数

#（3）将真实实验数据存储于一维数组中
x1 = np.array([20.5,32.7,51.0,73.0,95.7], dtype=float)      #温度
y1 = np.array([765, 826, 873, 942, 1032], dtype=float)      #电阻值

#（4）调用最小二乘拟合函数leastsq()
r = leastsq(residuals, [1,1], args=(x1, y1))    #返回值r[0]为所求解的拟合参数a，b
```

# 【例12.0】Python程序（续2）

```
#（5）输出拟合参数。数组r[0]存储的是拟合参数，r[1]、r[2]代表其他信息
a,b=r[0]
print('拟合参数a=%0.3f, b=%0.3f' % (a,b))       #保留小数点后三位

#（6）计算60度时的R值
R=a*60+b
print('60度时的R值为%0.3f' % R)

#（7）验证拟合参数的正确性
R_at20_5=a*20.5+b #计算t=20.5时的R值
error1=abs((R_at20_5-y1[0])/y1[0])*100          #计算相对误差
print('t=20.5度时的R值为%0.3f，相对误差为%1.2f%%' % (R_at20_5,error1))
R_at95_7=a*95.7+b
……
```

转换为百分比

“%%”表示打印“%”

# 【例12.0】Python程序运行结果

```
>>>
拟合参数a=3.399, b=702.097
60度时的R值为906.021
t=20.5度时的R值为771.771, 相对误差为0.89%
t=95.7度时的R值为1027.356, 相对误差为0.45%
>>>
```

**实验数据：**

t=20.5时，R=765

t=95.7时，R=1032

- **方法二：采用多项式拟合，利用NumPy库的多项式拟合函数polyfit()函数求解多项式系数a、b**

例12.0-thermistor-poly_fit.py

# 最小二乘法多项式拟合

- 在线性最小二乘拟合中，最简单、最常用的是采用**多项式拟合**
- 如果有一组实验观测数据，其拟合模型为多项式$f(x) = a_n x^n + a_{n-1}x^{n-1} \ldots + a_1 x + a_0$。寻找一组多项式系数$a_i$（$i = 0,1, \ldots \ldots, n$），使得 $f(x)$ 能够较好地拟合原始数据，使整体拟合误差较小。这种方法称为**多项式拟合**
  - **特例**：n=1，**一次多项式拟合**：**$f(x)=a_1x+a_0$**

# 多项式拟合函数polyfit()

- 对于多项式拟合，可以使用**NumPy**库的多项式拟合函数**polyfit()函数**求解多项式系数$a_i$（i=0~n）

```
import numpy as np
p=np.polyfit(x, y, n)
```

- ✓ 其中输入参数**x,y**为要拟合的数据，为**序列类型**（列表，元组或数组）；**n**为拟合多项式的次数
- ✓ polyfit( )函数的**返回值**为拟合多项式**系数**，**数组**形式

**注意：p的第一个元素对应多项式的最高次幂的系数（$a_n$），最后一个元素对应多项式的常数项（$a_0$）**

# 多项式求值函数polyval()

- 求得拟合多项式以后，可以使用**NumPy**库的多项式求值函数**polyval()**计算多项式在任意x处的值y

```
y=polyval(p,x)
```

- ✓ 参数**p**为**polyfit( )函数**返回的拟合多项式系数，**数组**形式
- ✓ 参数**x**为任意一个x坐标值，可以是**一个**值，也可以是**一组**值（存储于**列表**、**元组**或**数组**中）
- ✓ polyval()函数的返回值为该多项式在x处的值y（**数组**）

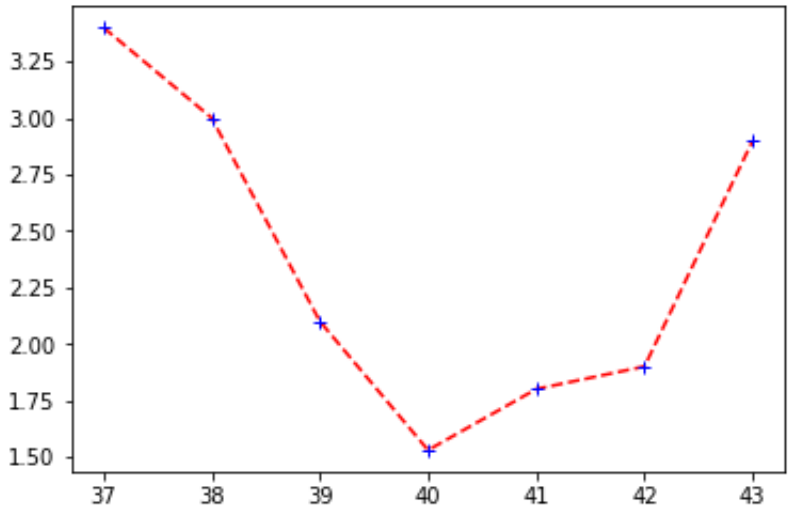
# 【引例12.2】合金成分与膨胀系数关系问题

- **【引例12.2】**某合金成分x与膨胀系数y之间的关系有如下表所示的实验数据。试求膨胀系数y与成分x的**近似函数关系**$y=f(x)$。并推测合金成分为**45**时的膨胀系数。

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 合金成分x | 37 | 38 | 39 | 40 | 41 | 42 | 43 |
| 膨胀系数y | 3.40 | 3.00 | 2.10 | 1.53 | 1.80 | 1.90 | 2.90 |

# 【引例12.2】求解思路

- **(1)** 根据表中实验数据**绘制散点图**和原始数据曲线
  - 观察数据分布规律，可以看出，它们大致分布在一条**抛物线**上




- **(2) 确定拟合模型**
  - 为此取拟合函数模型为$f(x) = a_2x^2 + a_1x + a_0$

# 【引例12.2】求解思路（续）

- **（3）求拟合参数**
  - 使用**NumPy**库的多项式拟合函数**polyfit( )函数**求解多项式系数$a_i$（i=0,1,2）
  - p=np.**polyfit**(x1,y1,2)，其中**2**为**拟合多项式的次数**
- **（4）求任意一点的值**
  - 使用**NumPy**库的多项式求值函数**polyval()**计算多项式在x=45处的值y，即合金成分为45时的膨胀系数

# 【引例12.2】 的Python程序

**引例12.2-膨胀系数-拟合-polyfit【方法一】.py**

```
#方法一 用polyfit()函数拟合
#导入三方库
import numpy as np
import matplotlib.pyplot as plt
from pylab import mpl          #为使图形正常显示中文，需要导入此模块

#（1）绘制原始数据点和原始数据曲线
x1 = np.arange(37, 44, 1)      #一组测量数据，合金成分，步长为1
y1 = np.array([3.40, 3.00, 2.10, 1.53, 1.80, 1.90, 2.90], dtype=float) #膨胀系数
plt.plot(x1,y1,"r--",marker="+",markeredgecolor="blue",label="原始数据")
#绘制原始数据曲线
```

# 【引例12.2】的Python程序（续1）

```
#（2）调用polyfit函数，求出拟合参数
p=np.polyfit(x1,y1,2)      #二次多项式：y=p[0]*x**2+p[1]*x+p[2]。p为数组
a2, a1, a0 = p             #p[0]对应a2，p[2]对应多项式的常数项a0
print('拟合参数：a2=%0.3f, a1=%0.3f, a0=%0.3f' % (a2, a1, a0 ))      #保留小数点后三位
print('拟合模型为：','y=%.3f*x**2+(%.3f)*x+(%.3f)' %(a2, a1, a0))

#（3）调用polyval函数，计算合金成分为45时的膨胀系数
x0=45                      插值
Result=np.polyval(p,x0)          #计算插值
#Result=a2*x0**2 + a1*x0+a0 #将x0直接带入拟合模型，效果同使用polyval函数
print('合金成分为45时的膨胀系数为%0.3f' % Result)
```

# 【引例12.2】的Python程序（续2）

```
#（4）绘制拟合曲线
#生成绘制拟合曲线所需的数据（插值）
x2=np.linspace(37, 43, 50)                #产生区间[37, 43]内等间隔的50个值
y2=np.polyval(p,x2)                       #使用polyval()函数计算多项式在x2处的值y2
plt.plot(x2,y2,'g-',label='拟合曲线')  #绿色实线

#（5）验证拟合参数的正确性
x_37=37
y_37=np.polyval(p,x_37)
print('合金成分为37时的膨胀系数为%0.3f' % y_37)
……
```

# 【引例12.2】 的Python程序（续3）

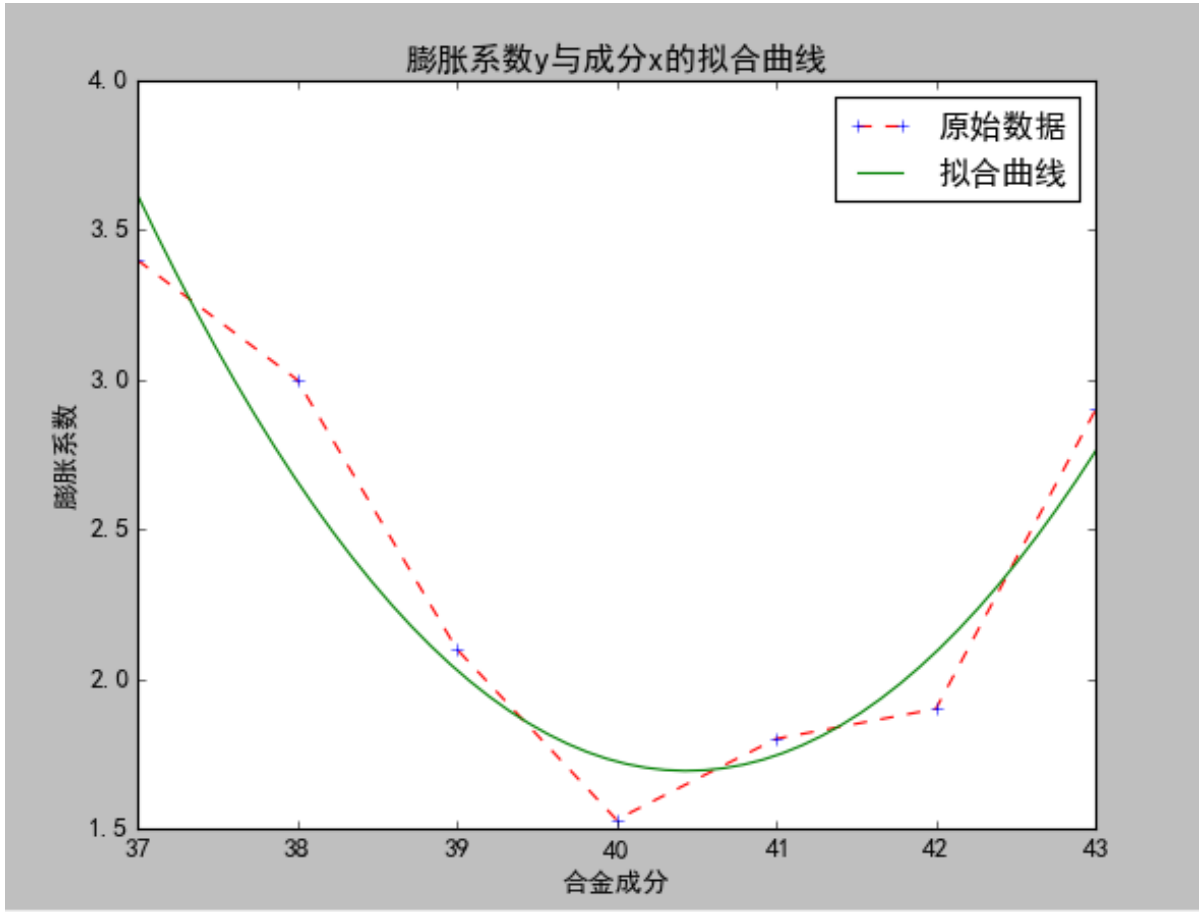
```
#美化图表
mpl.rcParams["font.sans-serif"] = ["SimHei"]   #指定默认字体为黑体
plt.title("膨胀系数y与成分x的拟合曲线")
plt.xlabel('合金成分')                          #设置x轴标签（x轴含义）
plt.ylabel('膨胀系数')                          #设置y轴标签（y轴含义）
plt.legend(loc="upper right")

plt.show()          #一定要放在最后！否则其后的print语句可能不执行
```

# 【引例12.2】程序运行结果

```
>>>
拟合参数a2=0.163, a1=-13.171, a0=268.010
拟合模型为： y=0.163*x**2+(-13.171)*x+(268.010)
合金成分为45时的膨胀系数为5.081
成分为37时的膨胀系数为3.619
成分为43时的膨胀系数为2.761
```

| 合金成分 | 37 | 43 |
|---|---|---|
| 膨胀系数实测值 | 3.40 | 2.90 |
| 膨胀系数计算值 | 3.619 | 2.761 |




- 计算结果与使用leastsq()函数（方法二）完全相同

# 12.2.4 非线性最小二乘拟合

$$f(x) = a_1 r_1(x) + a_2 r_2(x) + \cdots + a_m r_m(x)$$

- **非线性最小二乘拟合**：函数$f(x)$关于待定系数$\{a_k\}_{k=1}^{m}$是**非线性的**，则用这种函数形式作曲线的最小二乘拟合称为非线性最小二乘拟合。
- 非线性拟合函数，可以使用最小二乘拟合函数**leastsq()**求解其拟合参数
- 方法与使用leastsq()进行线性最小二乘拟合相同

# 常用的非线性拟合函数

- 非线性函数——**双曲线**$y = \frac{a_1}{x_1} + a_2$和**指数曲线**$y = a_1 e^{a2x}$，常用于非线性最小二乘拟合
- 过去在实际应用中，非线性拟合函数一般先通过**变量替换**转化为**线性函数**或**多项式函数**，使得求解拟合系数更简单
- 按**线性拟合**或**多项式拟合**求出待定系数后，再还原为原变量所表示的拟合模型

**注意：**现在使用**leastsq()函数**，可以直接进行非线性最小二乘拟合，不必对非线性拟合函数进行转换

# 【例12.5】非线性最小二乘拟合示例

【例12.5】体重约70kg的某人在短时间内喝下2瓶啤酒后，隔一定**时间t**测量他的血液中的**酒精含量**(mg/100ml)**h**，得到如下数据。试使用所给数据，用函数$f=at^be^{ct}$进行拟合，求出常数a,b,c。

t=[0.25,0.5, 0.75, 1, 1.5, 2, 2.5, 3, 3.5, 4,

4.5,5,6,7,8,9,10,11,12,13,14,15,16]

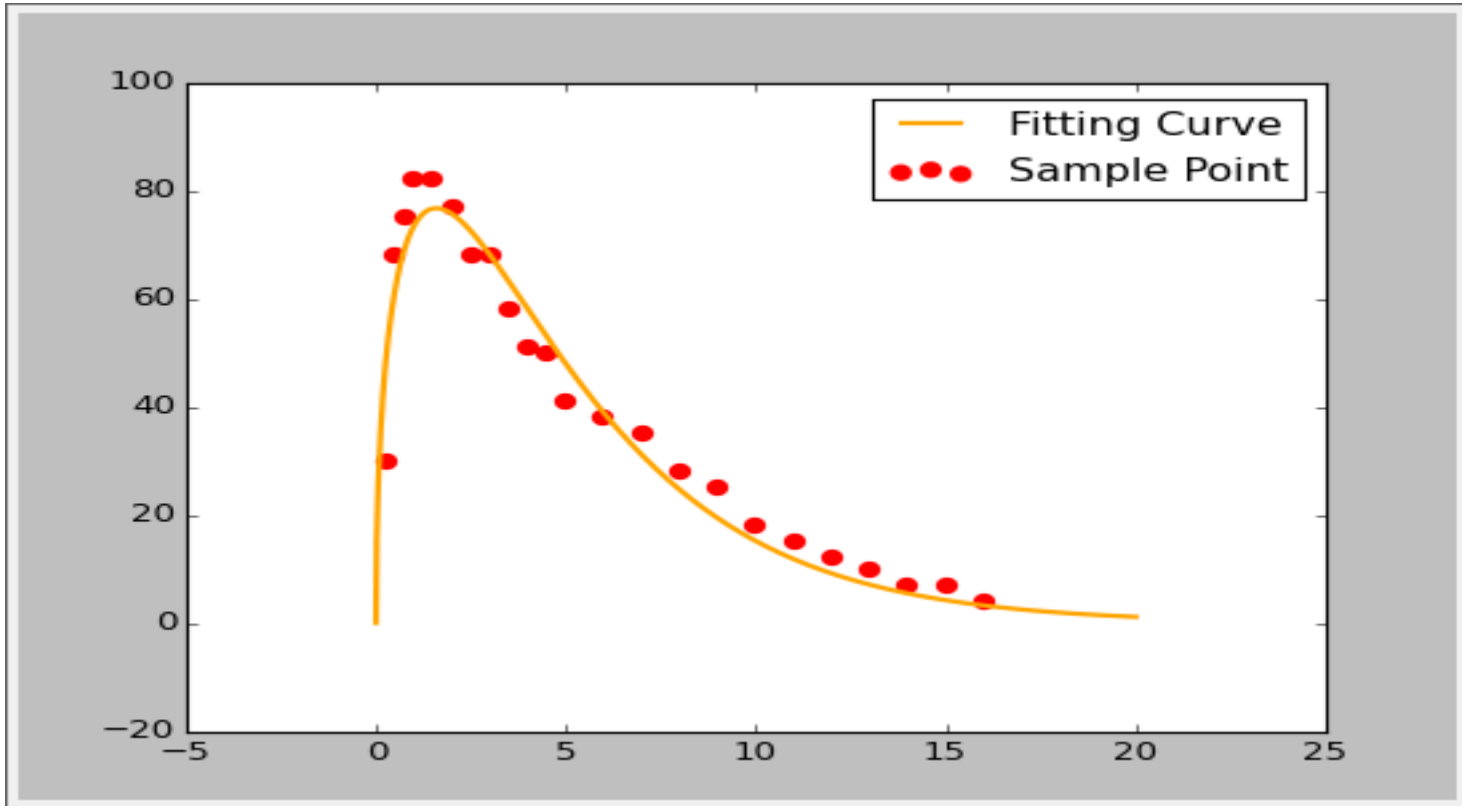
h=[30,68,75,82,82,77,68,68,58,51,50,41,38,35,28,25,18,15,12,10,7,7,4]

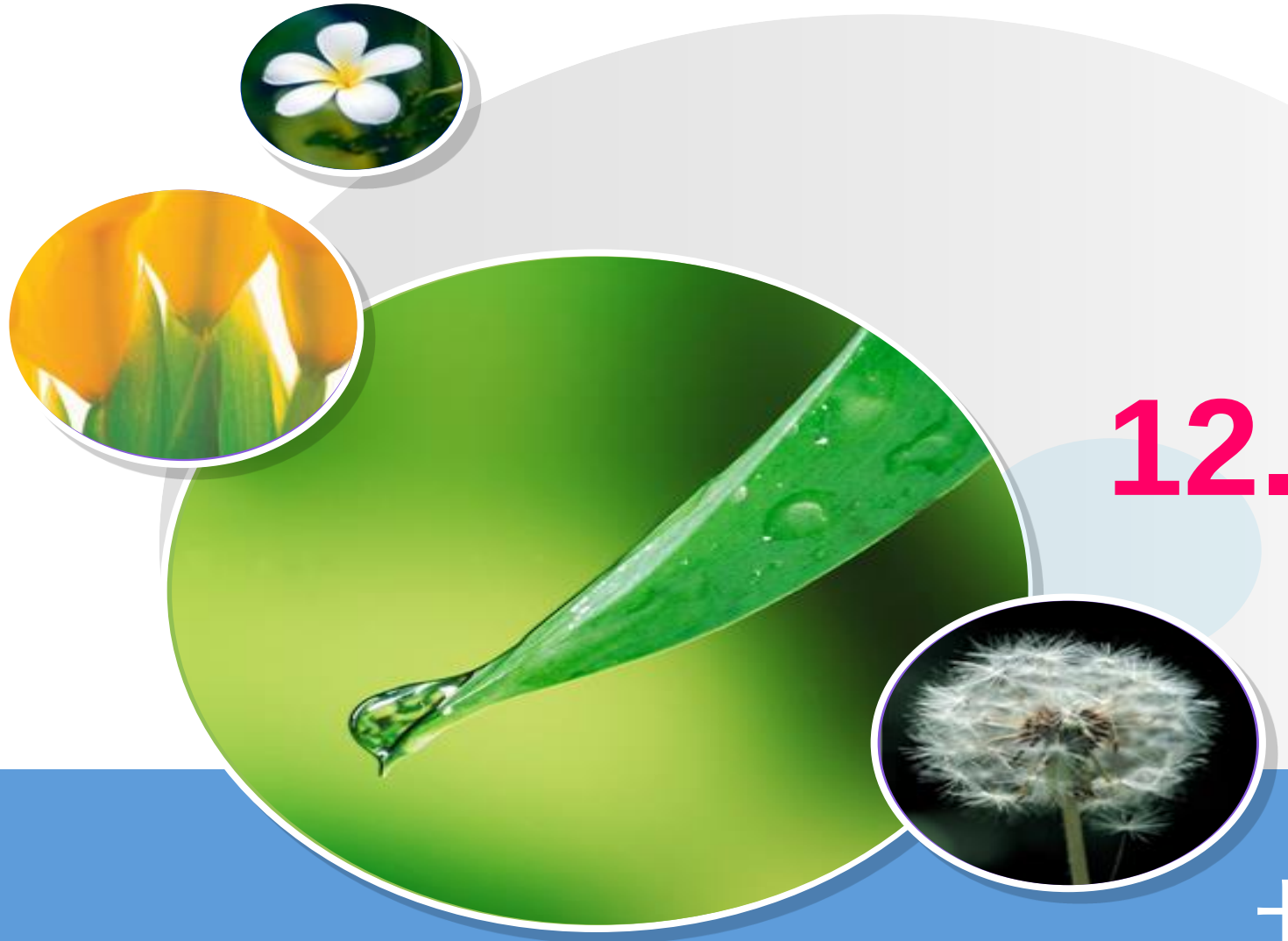
**课下学习、体会**

# 【例12.5 】程序运行结果

**例12.5-leastsq_fit.py**

```
>>>
拟合参数a=98.719, b=0.468, c=-0.295
```





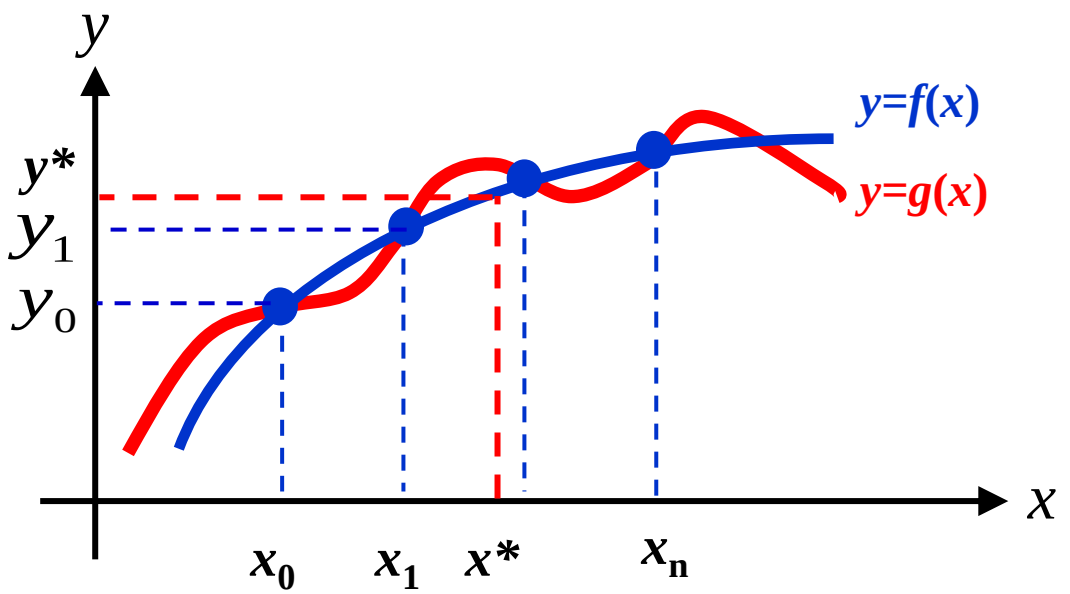
北京航空航天大學
BEIHANG UNIVERSITY

# 12.3 函数插值

北京航空航天大学

# 一维插值和二维插值

- **根据自变量的个数，插值分为一维插值和二维插值**
  - 只有一个自变量时，为一维插值，插值曲线经过二维平面上已知所有数据点
  - 有两个自变量时，为二维插值，插值曲面经过三维空间中已知所有数据点



# 常用的一维插值算法

- **最邻近插值**：用与被插值点x最相近节点的值作为x的值。得到的曲线**不光滑，精度不高**
- **分段线性插值**：用两点间的直线段（近似）代替未知的曲线段。**光滑性较差**
- **拉格朗日多项式插值**：构造**拉格朗日插值基函数**。**使用方便**，且**曲线光滑**

**缺点?**

- **牛顿插值，埃尔米特插值，三次样条插值**

- 当需要增加一个节点时，所有的基函数必须全部重新计算，不具备承袭性，会造成计算量的浪费
- 当插值多项式的次数超过7时，两端会出现严重的振荡现象（**龙格现象**）

# 常用的二维插值算法

- 常用的**二维插值算法**
  - **最邻近插值：**被插值点$(x^*, y^*)$的函数值等于与被插值点**最邻近**的节点的值。图像质量不高，易出现**锯齿边缘**
  - **分片线性插值：**在以相邻的4个网格节点为顶点的**小矩形**$R_{ij}$上**分两片**作线性插值。精度优于最邻近插值。
  - **双线性插值：**在以相邻的4个网格节点为顶点的**小矩形**$R_{ij}$上作**线性插值。**精度优于最邻近插值。

# 1. 一维插值函数interp1d()

- **SciPy库中的interpolate模块**提供了许多对数据进行**插值运算**的函数
- **一维插值函数interp1d()**
  - interp1d()函数进行**一维多项式**插值
  - 用多项式函数通过已知的**所有**数据点；计算目标插值点上的**插值**函数值

# interp1d()的用法

- **用法：interpolate.interp1d(x, y, kind=kind)**
  - ✓ 其中，**x**和**y**参数是一系列已知的数据点x、y采样值，**一维数组**
  - ✓ **kind**参数是**插值类型**，可以是**字符串**或**整数**，它给出插值的B样条曲线的阶数
  - ✓ 其**返回值**为一个**函数**（多项式插值函数），需要传递给它的参数为一维插值点的x值，则可以计算x相应的插值函数值

| kind的候选值 | 作用 |
| --- | --- |
| 'zero'、'nearest' | **阶梯**插值，**最邻近**插值，相当于0阶B样条曲线 |
| ‘linear’、'slinear' | **分段线性（linear）**插值（**默认**），**一阶**B样条插值（slinear）。**二者效果一样** |
| 'quadratic'、'cubic' | **二阶（quadratic）和三阶（cubic）B样条插值**，更高阶的插值可以直接使用整数值指定 |

# 【课后练习】

**【课后练习】**使用不同的插值方式（**“nearest”,“zero”, “linear”,“slinear”,“quadratic”,“cubic”**），对正弦波在区间（[0, 2π]上的点进行一维插值，比较它们的效果。**哪种插值效果最好?**

- **思考**：程序宜采用什么控制结构?
- **猜一猜**：哪种插值效果最好?

# 【引例12.1 】环境温度问题

- **【引例12.1 】**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数据分别为：12，9，9，10，18，24，28，27，25，20，18，15，13。试推测中午**1点**的温度，并作出24小时内的温度变化曲线图。

# 【引例12.1】求解思路之（1）

- **（1）绘制原始数据点及其曲线**
  - 使用NumPy库的**linspace**函数在[0,24]之间创建**一维数组**x，作为原始数据点x值（时间）
  - 使用NumPy库的**array**函数，将存储测量数据y值（温度）的列表转换为**一维数组**
  - 然后使用matplotlib.pyplot 模块的**plot**函数，绘制原始数据点及其形成的曲线

# 【引例12.1】求解思路之（2）

- **（2）一维插值，得到多项式插值函数**
  - 调用**interpolate**模块的**interp1d()**函数，指定其kind参数为“**cubic**”，即选择**三次样条插值**

```
f=interpolate.interp1d(x1,y1,kind="cubic")
```

# 【引例12.1】求解思路之（3）

## （3）绘制插值后曲线

- 绘制曲线，需要数据点**足够多**，才能绘制出**光滑**的曲线
- 使用NumPy库的**arange**函数在[0,24]之间创建一维数组**xnew**，生成一组点数更多的数据（取步长=0.5），作为插值数据点x值
- 将xnew传递给调用interp1d()后的**返回值f（插值函数）**，计算插值结果**ynew=f(xnew)**
- 然后绘制插值后曲线

# 【引例12.1】求解思路之（4）

- **（4）计算中午1点（即13点）的温度**
  - 将13传递给调用interp1d()后的返回值（实际为一个函数）

**T_at13=f(13)**

# 【引例12.1】 Python程序

**引例12.1-temperature-插值.py**

```
import numpy as np
from scipy import interpolate                    #导入插值模块
import matplotlib.pyplot as plt
from pylab import mpl                            #为了正常显示中文

#（1）绘制原始数据点及其曲线
x1 = np.linspace(0, 24, 13)                      #一组测量数据的x值（时间）
y1 = np.array([12,9,9,10,18,24,28,27,25,20,18,15,13], dtype=float)  #y值（温度）
plt.plot(x1,y1,"r--",marker="+",markeredgecolor="blue",label="原始数据")
```

离散数据点
形状和颜色

# 【引例12.1】 Python程序（续）

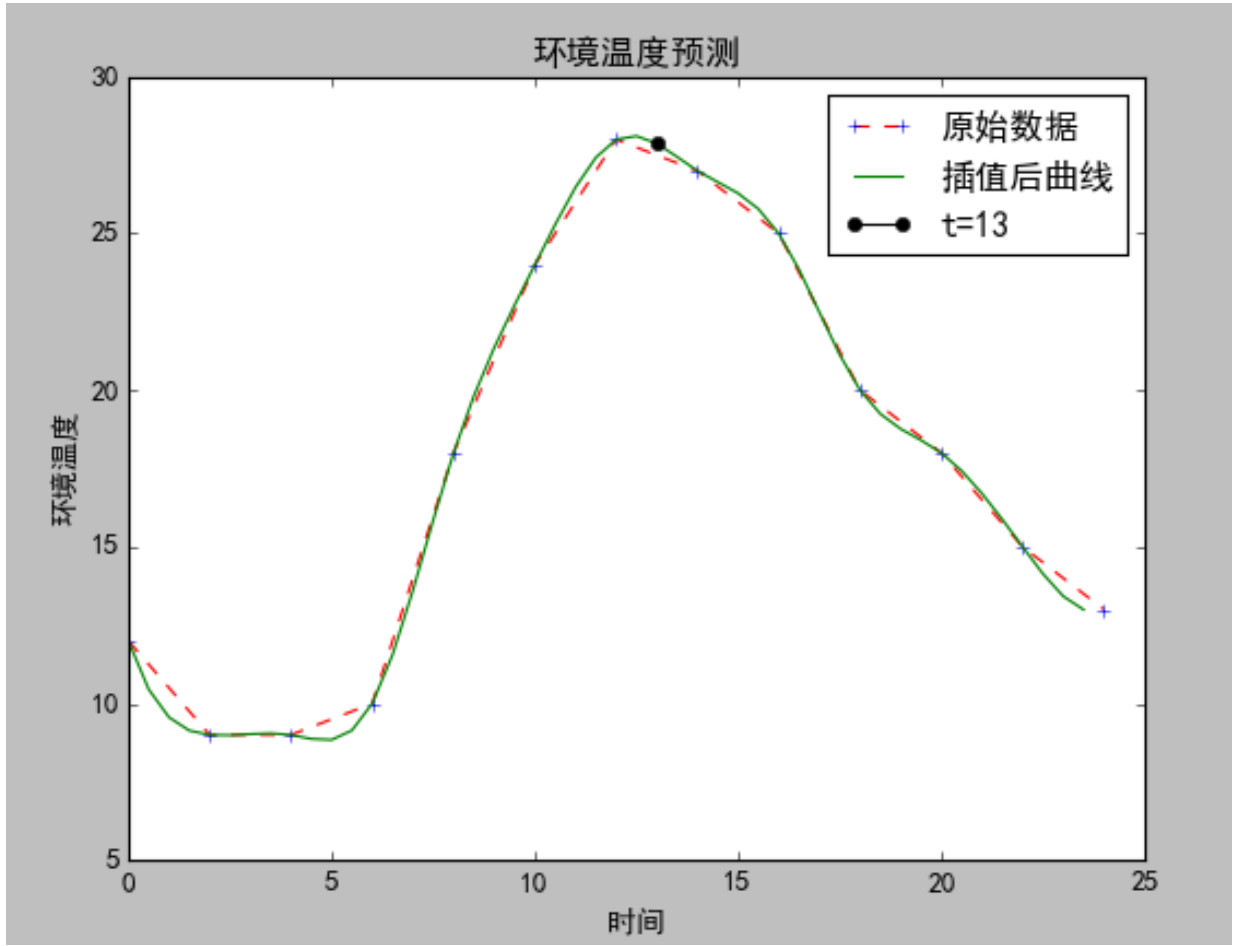
```
#（2）一维插值，得到多项式插值函数
f=interpolate.interp1d(x1,y1,kind="cubic")  #创建interp1d对象(三次样条插值)。返回值为一个函数

#（3）绘制插值后曲线
xnew = np.arange(0, 24, 0.5)                #建立插值数据点，一维数组
ynew=f(xnew)                                #计算插值结果
plt.plot(xnew,ynew,"g-",label="插值后曲线")  #绿色实线

#（4）计算中午1点（即13点）的温度
T_at13=f(13)
print('13点时的温度值为%0.2f' % T_at13)
plt.plot(13,T_at13,"k",marker="o",label="t=13") #绘制插值点
```

# 【引例12.1】程序运行结果

```
# (5) 美化图表
……
```




```
>>>
13点时的温度值为27.87
```

- 由于采用**三次样条**插值，所以插值曲线较好地逼近了原始数据曲线

# 2. 二维插值函数interp2d()

- **二维插值函数interp2d()**
  - interp2d()函数进行**二维多项式**插值
  - 该函数使用多项式技术，使**多项式函数**通过已知的**所有**数据点；计算目标插值点上的**插值**函数值

# interp2d()的用法

- **用法：** **interpolate.interp2d(x, y, fvals，kind=kind)**
  - ✓ 其中，**x**和**y**参数都是**多维数组（矩阵）**，存储曲面上一系列已知数据点的**x、y坐标值**
  - ✓ **fvals**是与已知数据点对应的**函数样本值**
  - ✓ 用**kind**指定的算法计算插值：
    - **linear**：双线性插值（**缺省**）
    - **nearest**：最邻近插值
    - **spline**：三次样条插值
    - **cubic**：双三次插值
  - ✓ 其**返回值**为一个**函数**（多项式插值函数），需要传递给它的参数为二维插值点的**x、y值**

二维数组存储

# 【例12.6】二维插值案例

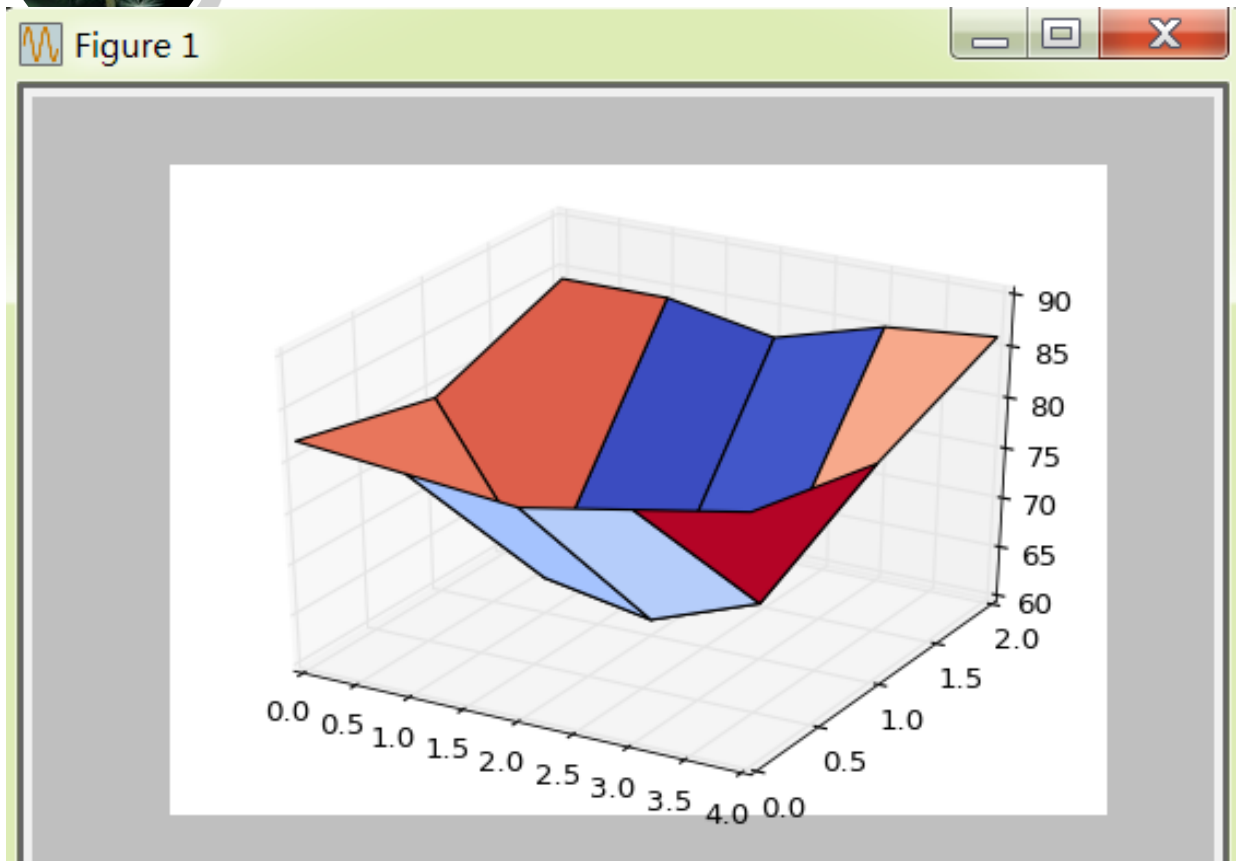
**【例12.6】**测得平板表面3*5（3行5列）网格点处的温度如下所示。试画出曲面z=*f*(x,y)的图形；在x-y平面上每0.1个单位的地方进行插值以平滑数据。

平板表面3*5网格点处的温度值

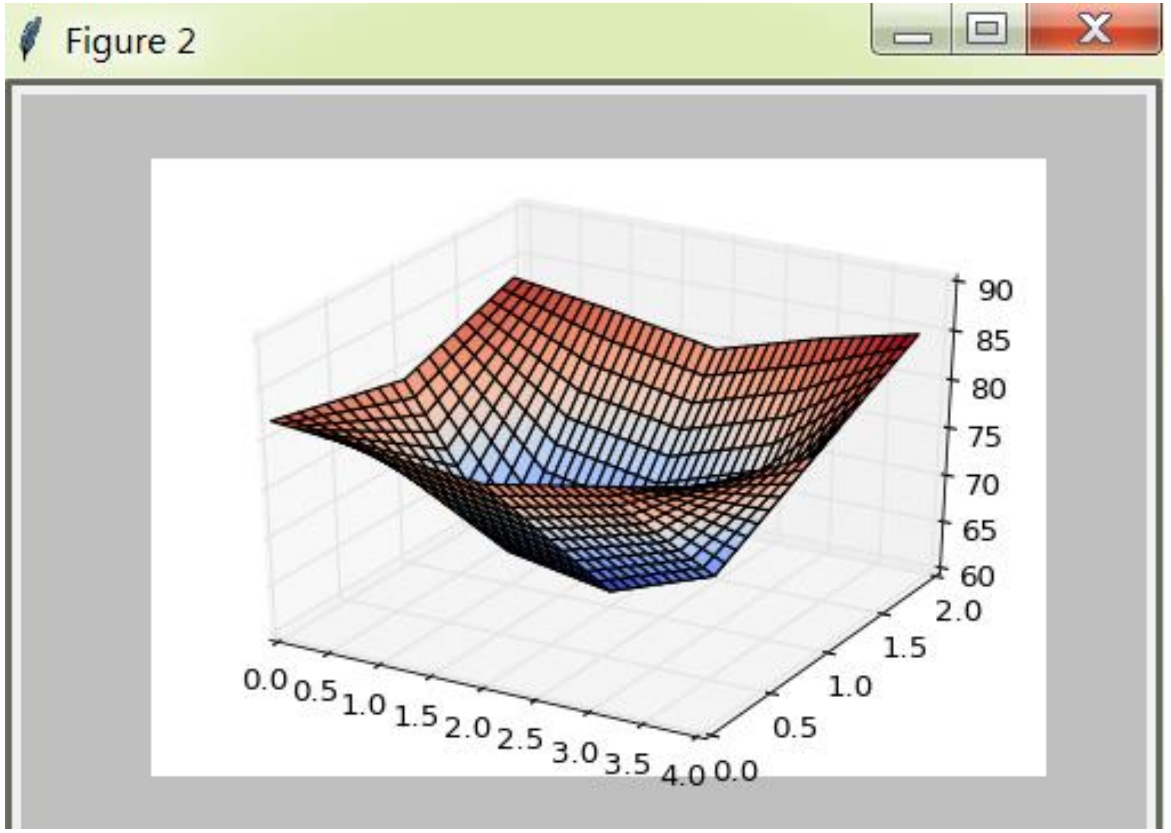
| | | | | |
|---|---|---|---|---|
| 82 | 81 | 80 | 82 | 84 |
| 79 | 63 | 61 | 65 | 81 |
| 84 | 84 | 82 | 85 | 86 |

**详见教材第5章【案例5.2】和12.3课件【例12.6 】**

# 【例12.6 】程序运行结果




原始平板温度分布网线图




插值后的平板温度分布网线图

**例12.6-二维插值案例.py**