实验一　　插值法

一、　算法设计思想:

在插值计算中，取样点的多少往往会影响所得插值函数优化程度。一般情况下，取样点越多所得插值函数越优化，对应的函数值与标准函数值越接近。

在 MATLAB 中实现分段线性插值，分段二次插值，拉格朗日插值的命令为 interp1,其调用格式为: Y1=interp1(X,Y,X1，’method’)

函数根据 X,Y 的值，计算函数在 X1 处的值。X,Y 是两个等长的已知向量，分别描述采样点和样本值，X1 是一个向量或标量，描述欲插值点，Y1 是一个与 X1 等长的插值结果。

二、　程序清单：

1. 分段线性插值

```
#include<stdio.h>
#include<math.h>
 double Lagrange2(double *x, double *y, double input)
{

 double output;
 int i;

 for (i=0;i<5;i++)
 {
  if (x[i] <= input && x[i+1] >= input)
  {
   output=y[i] +(y[i+1]-y[i])*(input-x[i])/(x[i+1]-x[i]);
   break;
  }
 }
 return output;
}
```

2.分段二次插值

```
double Lagrange3(double *x,double *y,double u)
{
 int i,k=0;
 double v;
 for(i=0;i<6;i++)
 {
  if(u<x[1])
  {
   k=0;

v=y[k]*(u-x[k+1])*(u-x[k+2])/((x[k]-x[k+1])*(x[k]-x[k+2]))+y[k+1]*(u-
```

```
x[k])*(u-x[k+2])/((x[k+1]-x[k])*(x[k+1]-x[k+2]))+y[k+2]*(u-x[k])*(u-x
[k+1])/((x[k+2]-x[k])*(x[k+2]-x[k+1]));
  }
  if((x[i]<u&&u<=x[i+1])&&(fabs(u-x[i])<=fabs(u-x[i+1])))
  {
   k=i-1;

v=y[k]*(u-x[k+1])*(u-x[k+2])/((x[k]-x[k+1])*(x[k]-x[k+2]))+y[k+1]*(u-
x[k])*(u-x[k+2])/((x[k+1]-x[k])*(x[k+1]-x[k+2]))+y[k+2]*(u-x[k])*(u-x
[k+1])/((x[k+2]-x[k])*(x[k+2]-x[k+1]));
  }
  if ((x[i]<u&&u<=x[i+1])&&fabs(u-x[i])>fabs(u-x[i+1]))
  {
   k=i;

v=y[k]*(u-x[k+1])*(u-x[k+2])/((x[k]-x[k+1])*(x[k]-x[k+2]))+y[k+1]*(u-
x[k])*(u-x[k+2])/((x[k+1]-x[k])*(x[k+1]-x[k+2]))+y[k+2]*(u-x[k])*(u-x
[k+1])/((x[k+2]-x[k])*(x[k+2]-x[k+1]));
  }
  if(u>x[4])
  {
   k=3;

v=y[k]*(u-x[k+1])*(u-x[k+2])/((x[k]-x[k+1])*(x[k]-x[k+2]))+y[k+1]*(u-
x[k])*(u-x[k+2])/((x[k+1]-x[k])*(x[k+1]-x[k+2]))+y[k+2]*(u-x[k])*(u-x
[k+1])/((x[k+2]-x[k])*(x[k+2]-x[k+1]));
  }
    }
 return v;
}
void main()
{
 double x[6] = {0.0, 0.1, 0.195, 0.3, 0.401, 0.5},y[6] =
{0.39894,0.39695,0.39142,0.38138,0.36812,0.35206};
 double u;
 scanf("%lf",&u);
```

拉格朗日插值：

```
double Lagrange1(double *x, double *y, double xx)
{
 int i,j;
 double *a,yy=0.000;
 a=new double[6];
```

```
 for(i=0;i< 6;i++)
 {
  a[i]=y[i];
  for(j=0;j< 6;j++)
   if(j!=i)
    a[i]*=(xx-x[j])/(x[i]-x[j]);
   yy+=a[i];
 }
 delete a;
 return yy;
}
double Lagrange2(double *x, double *y, double input)
{

 double output;
 int i;

 for (i=0;i<5;i++)
 {
  if (x[i] <= input && x[i+1] >= input)
  {
   output=y[i] +(y[i+1]-y[i])*(input-x[i])/(x[i+1]-x[i]);
   break;
  }
 }
 return output;
}
```

三、 运行结果：

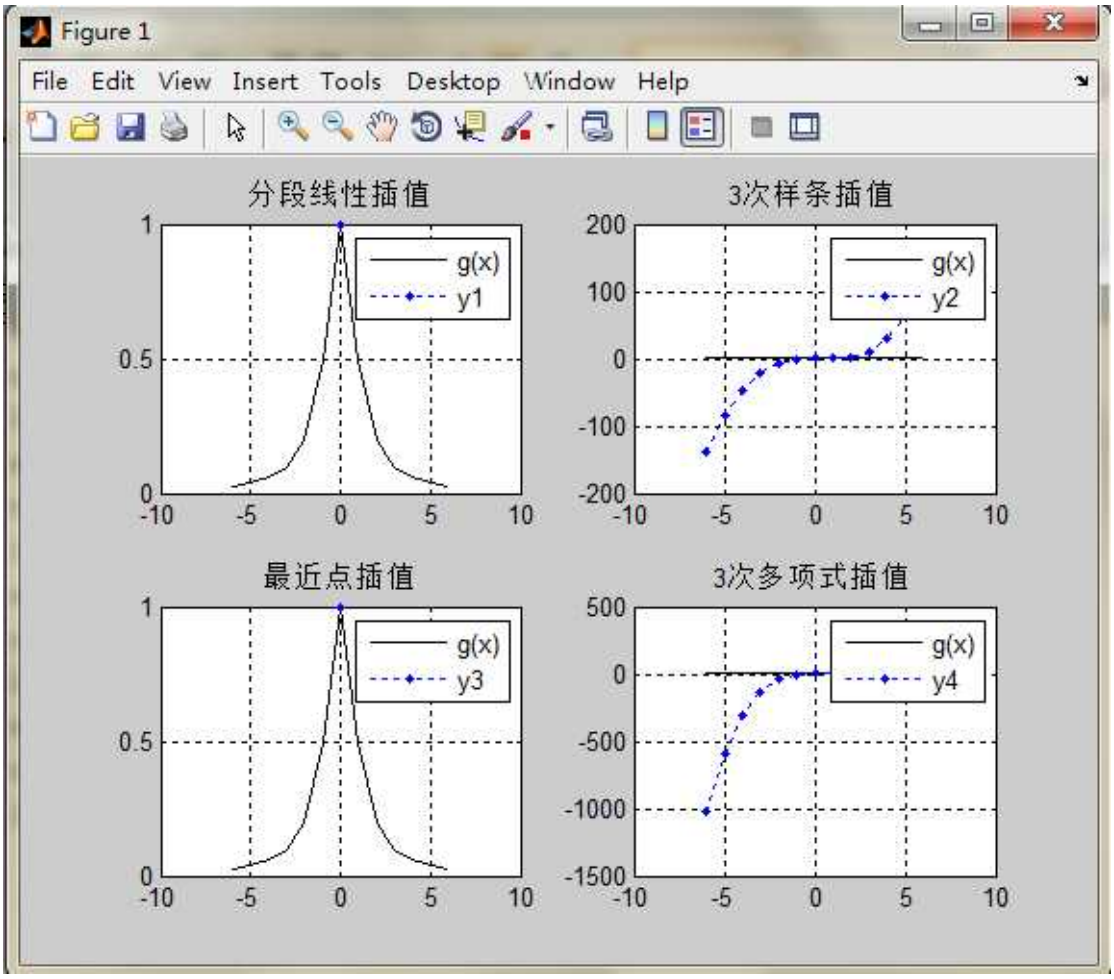


Figure 1
File Edit View Insert Tools Desktop Window Help
分段线性插值
g(x)
y1
3次样条插值
g(x)
y2
最近点插值
g(x)
y3
3次多项式插值
g(x)
y4

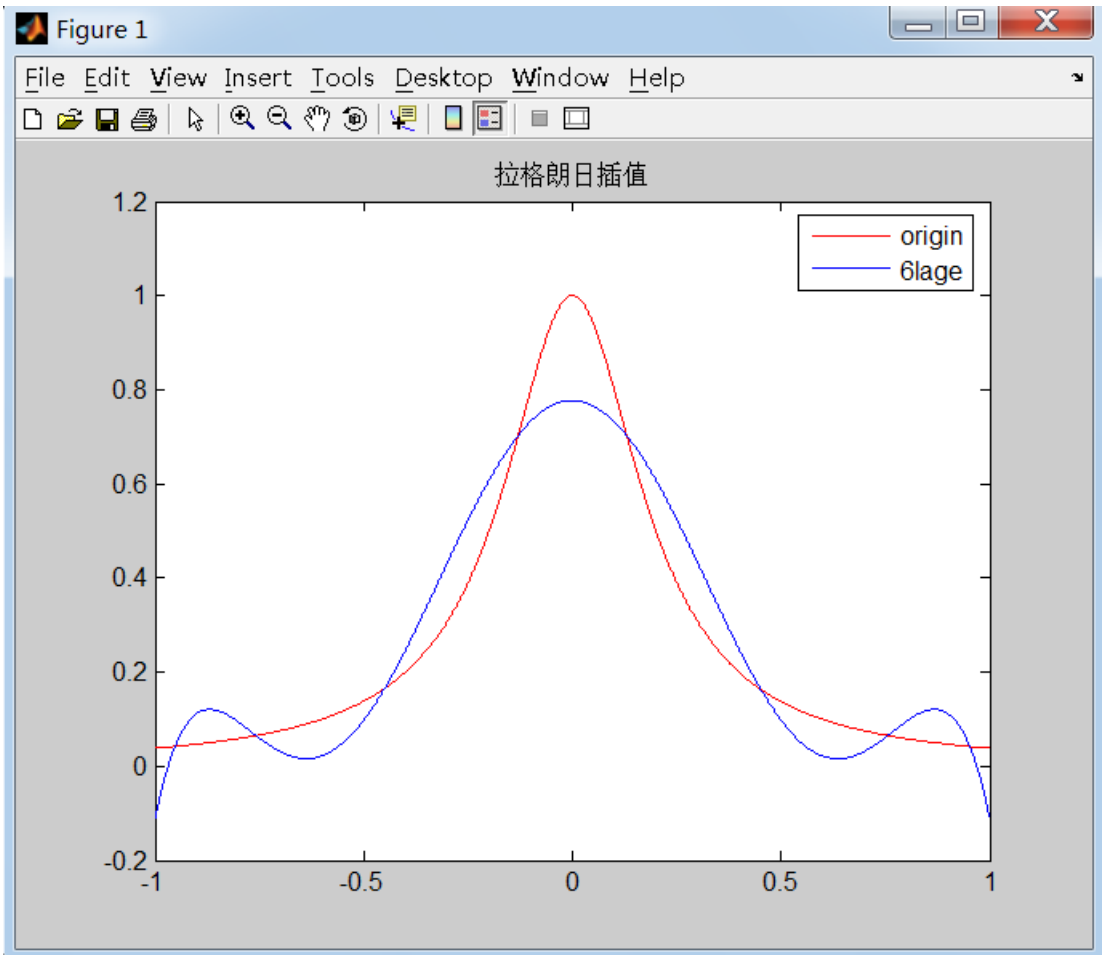
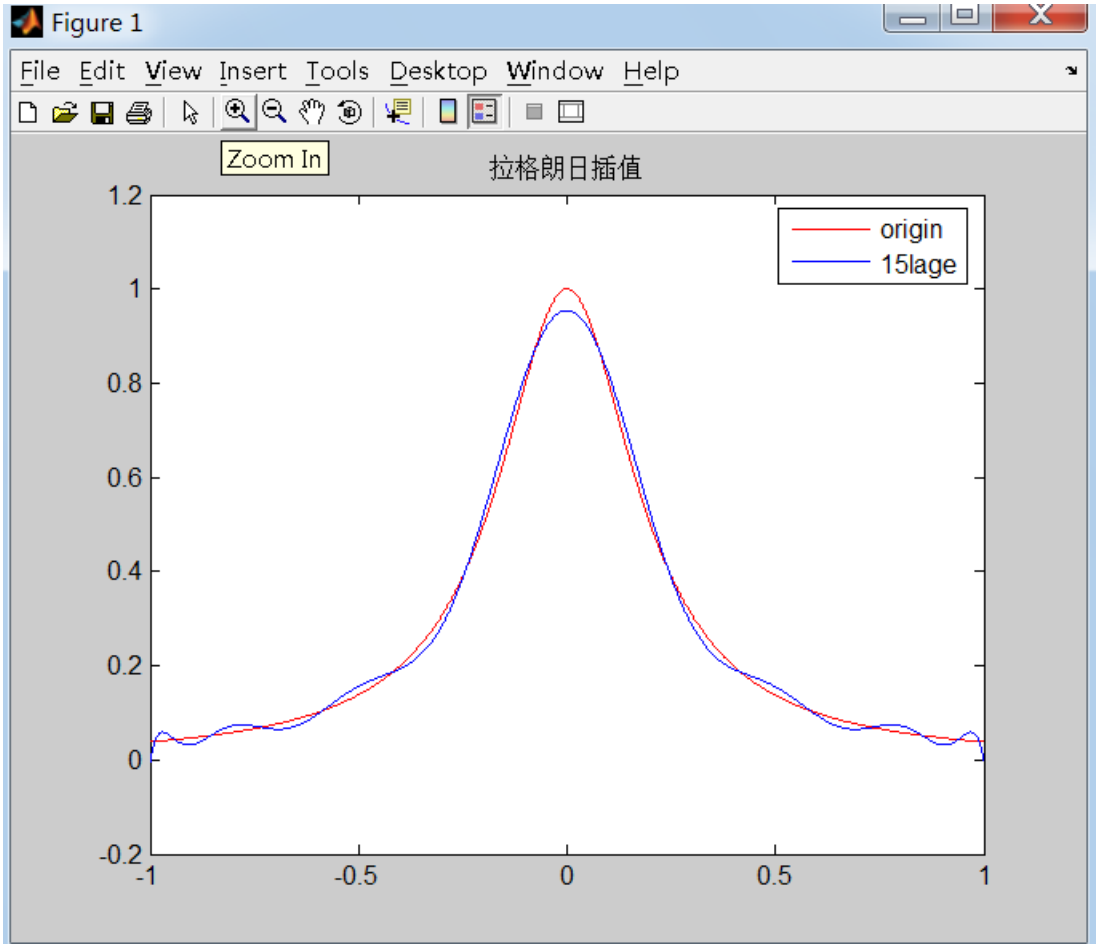


Figure 1
File Edit View Insert Tools Desktop Window Help
拉格朗日插值
origin
6lage
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-1
-0.5
0
0.5
1



四、　四种插值的比较：

从以上对比函数图象可以看出，分段线性插值其总体光滑程度不够。在数学上，光滑程度的定量描述是函数(曲线）的 k 阶导数存在且连续，则称该曲线具有 k 阶光滑性。一般情况下，阶数越高光滑程度越好。分段线性插值具有零阶光滑性，也就是不光滑。二次插值就是较低次数的多项式而达到较高阶光滑性的方法。总体上分段线性插值具有以下特点：

五、经验总结：

分段线性插值在计算上具有简洁方便的特点。分段线性插值与二次插值函数在每个小区间上相对于原函数都有很强的收敛性，（舍入误差影响不大），数值稳定性好且容易在计算机上编程实现等优点，但分段线性插值在节点处具有不光滑性的缺点(不能保证节点处插值函数的导数连续)，从而不能满足某些工程技术上的要求。

实验二　曲线拟合的最小二乘法

一、　算法设计思想：

对给定数据　　(i=0 ,1,2,3,..,m),一共 m+1 个数据点,取多项式 P(x),使

$$\sum_{i=0}^{m} r_i^2 = \sum_{i=0}^{m} [p(x_i) - y_i]^2 = \min$$

函数 P(x)称为拟合函数或最小二乘解，令 $p_n(x) = \sum_{k=0}^{n} a_k x^k$ 使得

$$I = \sum_{i=0}^{m} [p_n(x_i) - y_i]^2 = \sum_{i=0}^{m} \left(\sum_{k=0}^{n} a_k x_i^k - y_i \right)^2 = \min$$

其中，a0，a1,a2,…,an 为待求未知数，n 为多项式的最高次幂，由此，该问题化为求

的极值问题。由多元函数求 $I = I(a_0, a_1, \cdots a_n)$

n

得到：

$$\frac{\partial I}{\partial a_j} = 2\sum_{i=0}^{m} (\sum_{k=0}^{n} a_k x_i^k - y_i) x_i^j = 0$$

$$\sum_{k=0}^{n} (\sum_{i=0}^{m} x_i^{j+k}) a_k = \sum_{i=0}^{m} x_i^j y_i,$$, n

这是一个关于 a0，a1,a2,…,an 的线性方程组，用矩阵表示如下：

$$\begin{bmatrix} m+1 & \sum_{i=0}^{m} x_i & \cdots & \sum_{i=0}^{m} x_i^n \\ \sum_{i=0}^{m} x_i & \sum_{i=0}^{m} x_i^2 & \cdots & \sum_{i=0}^{m} x_i^{n+1} \\ \vdots & \vdots & & \vdots \\ \sum_{i=0}^{m} x_i^n & \sum_{i=0}^{m} x_i^{n+1} & \cdots & \sum_{i=0}^{m} x_i^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{m} y_i \\ \sum_{i=0}^{m} x_i y_i \\ \vdots \\ \sum_{i=0}^{m} x_i^n y_i \end{bmatrix}$$

因此，只要给出数据点 (x_i, y_i) 及其个数 m，再给出所要拟合的参数 n，则即可

求出未知数矩阵（a0，a1,a2,…,an）

二、程序清单：

```
x=-1.0:0.5:2.0;y=[-4.447,-0.452,0.551,0.048,-0.447,0.549,4.552];
         plot(x,y,'*')
         xlabel 'x 轴'
         ylabel 'y 轴'
         title '散点图'
         hold on

         m=6;n=3;
         A=zeros(n+1);
         for j=1:n+1
        for i=1:n+1
        for k=1:m+1
        A(j,i)=A(j,i)+x(k)^(j+i-2)
           end
          end
         end;

       B=[0 0 0 0];
       for j=1:n+1
        for i=1:m+1
          B(j)=B(j)+y(i)*x(i)^(j-1)
      end
  end
 B=B';
 a=inv(A)*B;
 x=[-1.0:0.0001:2.0];
z=a(1)+a(2)*x+a(3)*x.^2+a(4)*x.^3;
plot(x,z)
legend('离散点','y=a(1)+a(2)*x+a(3)*x.^2+a(4)*x.^3')
title('拟合图')
```

三、运行的结果：

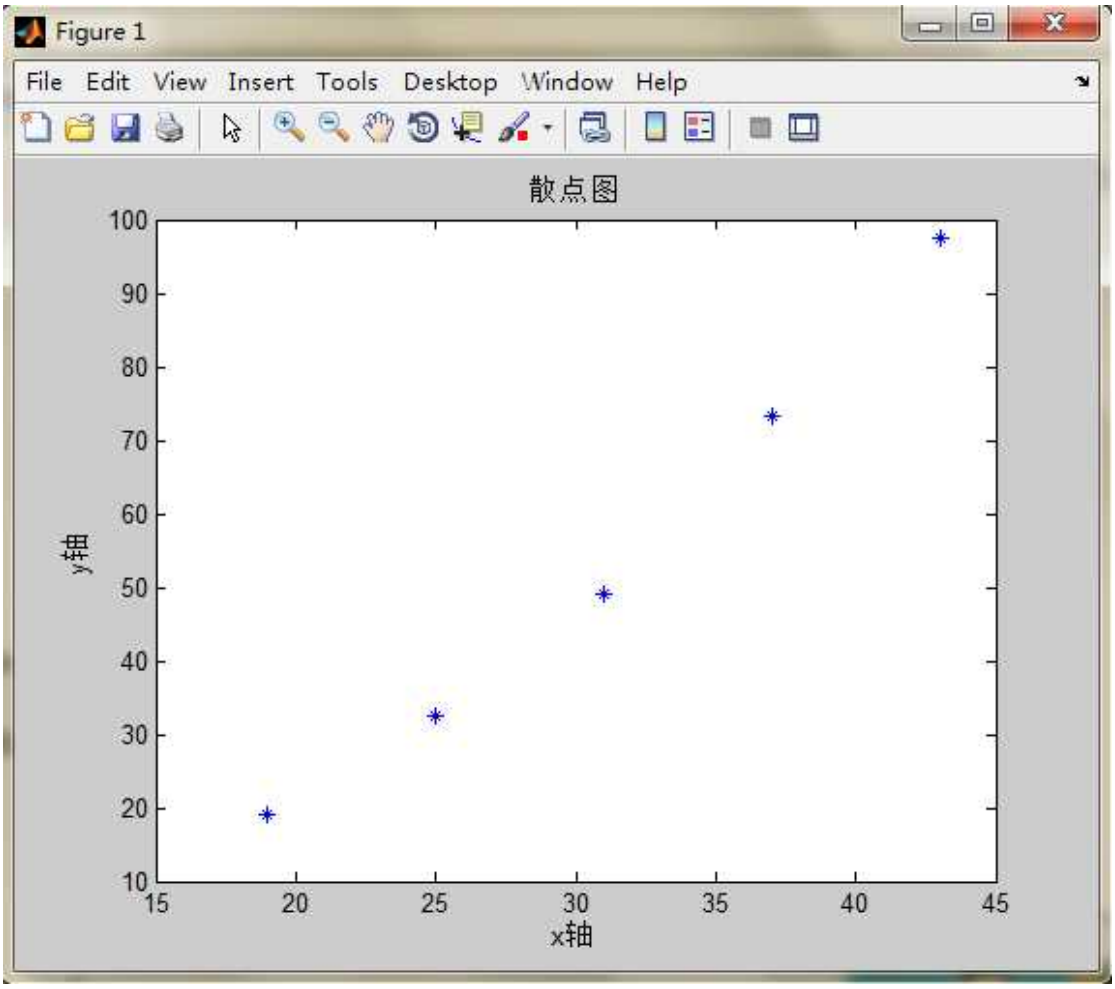


Figure 1
File Edit View Insert Tools Desktop Window Help
散点图
y轴
x轴
100
90
80
70
60
50
40
30
20
10
15
20
25
30
35
40
45

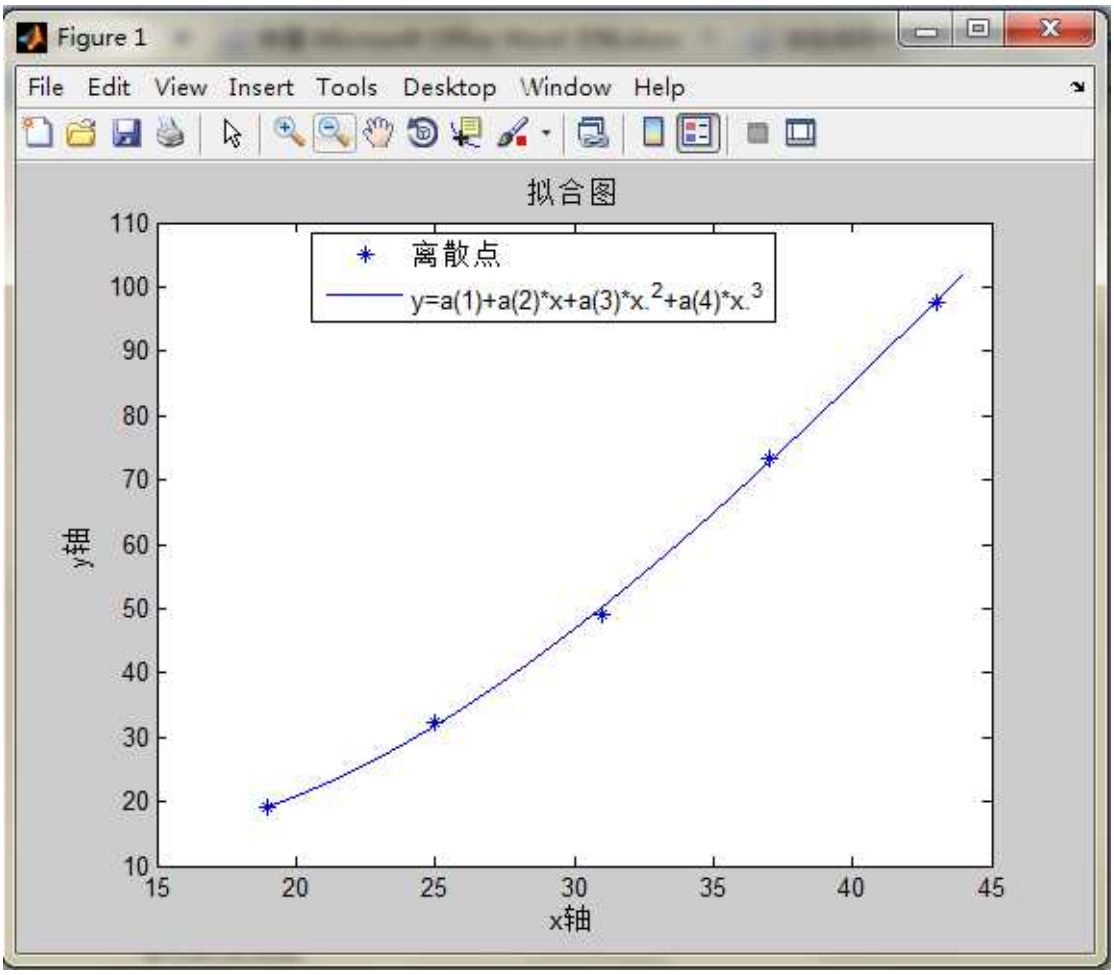


Figure 1
File Edit View Insert Tools Desktop Window Help
拟合图
离散点
y=a(1)+a(2)*x+a(3)*x.^2+a(4)*x.^3
y轴
x轴
110
100
90
80
70
60
50
40
30
20
10
15
20
25
30
35
40
45

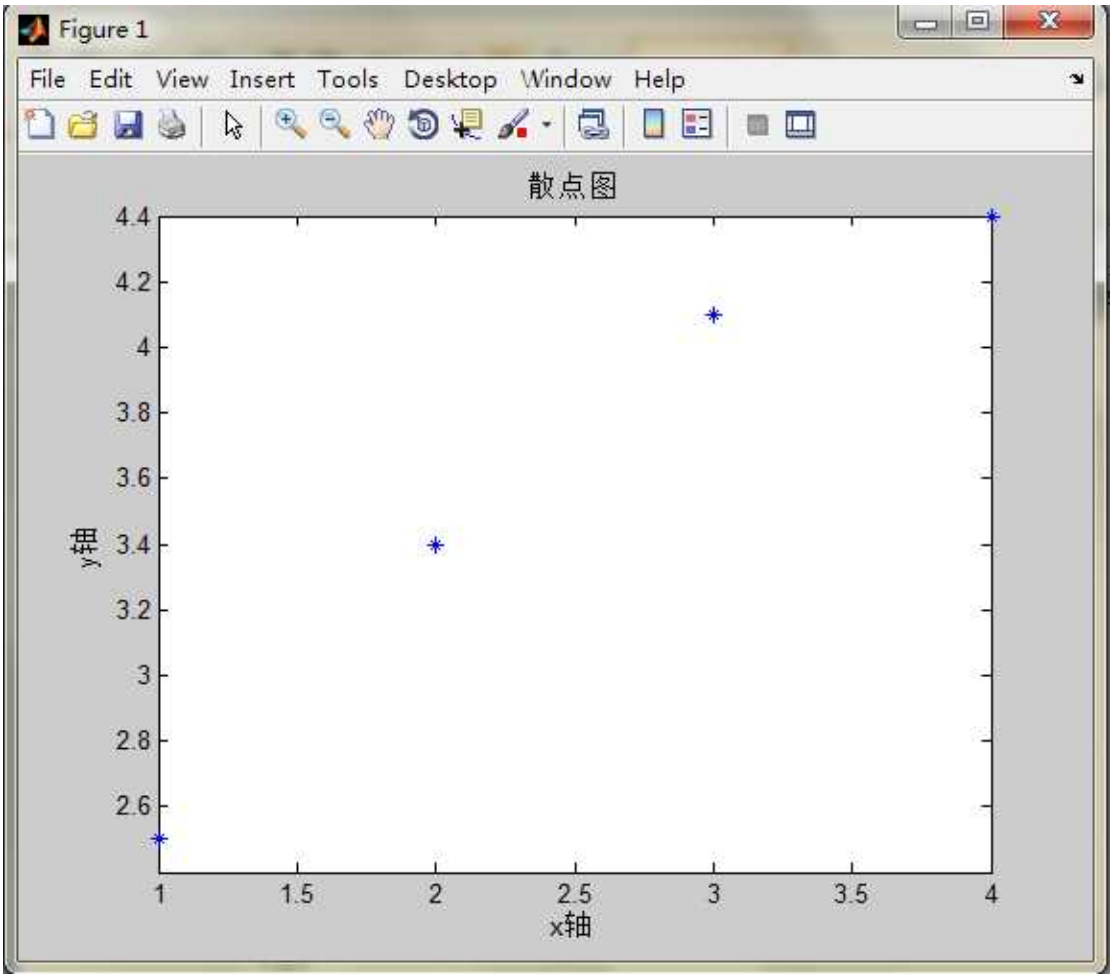


Figure 1
File Edit View Insert Tools Desktop Window Help
散点图
4.4
4.2
4
3.8
3.6
3.4
3.2
3
2.8
2.6
y轴
1
1.5
2
2.5
3
3.5
4
x轴