第四章报告

问题描述：由于钢水对材料的侵蚀作用，随着使用次数的增加，炼钢厂所用的盛钢水的钢包的容积不断增大。现希望获得增大容积 y 与使用次数 x 之间的函数关系，实测得到如下数据：

使用次数 x	2	3	4	5	6	7	8	9
增大容积 y	6.7	8.2	9.58	9.5	9.7	10	9.96	9.99
使用次数 x	10	11	12	13	14	15	16	
增大容积 y	10.49	10.59	10.6	10.8	10.6	10.9	10.76	

（1）请用插值方法建立 y 与 x 之间的函数关系，画出散点图和插值函数曲线，从结果说明插值方法是否适合该问题

（2）请分别从 1. $\frac{1}{y}=a+\frac{b}{x}$ 和 2. $y=ae^{\frac{b}{x}}$ 两种形式拟合建立 y 与 x 之间的函数关系，画出散点图和拟合函数曲线，并根据你选定的适当指标判断哪一种形式更好。

问题分析：

（1）插值方法有 Lagrange 插值多项式法，Newton 插值多项式法，分段低次插值和样条插值法。

（2）本题目中已知了 15 组数据，使用低次 Langrange 或 Newton 插值会与理想中的结果产生较大的偏差。提高插值多项式的次数，可以提高计算结果的准确程度，但随着 n 的增大，也会产生龙格现象。本题中 n>7，不宜采用高次多项式插值，而采用分段低次插值、样条插值或用次数较低的最小二乘逼近。

Matlab 程序组织结构：

在主程序 main.m 中运行各函数程序

函数：Interpolation_Split1　分段一次拉格朗日插值
Interpolation_Split2　分段二次拉格朗日插值
Interpolation_Spine2　二次样条插值
Interpolation_Spine3　三次样条插值
Lagrange_1　一次拉格朗日插值
Lagrange_2　二次拉格朗日插值
Newton　高次牛顿插值
fit　最小二乘法拟合曲线的系数计算
myplot　我的绘图函数，可同时计算均方误差
Guass_Jordan　高斯约旦法求解线性方程组

Matlab 程序：

一、低次拉格朗日插值

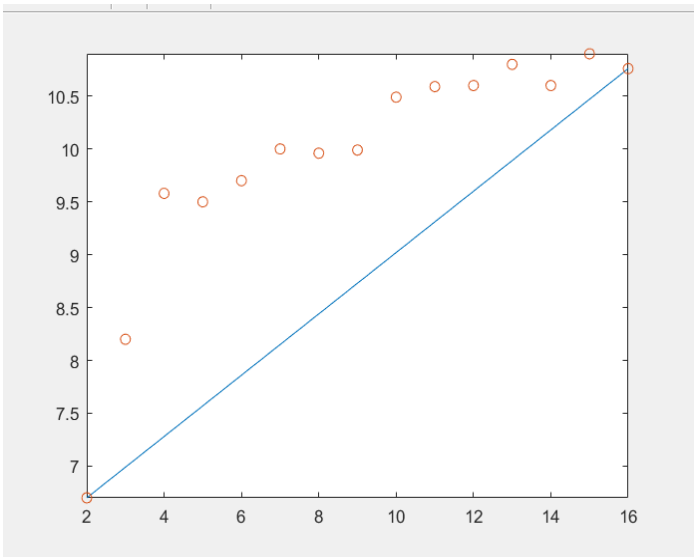
传入参数为两个点的 x 与 y

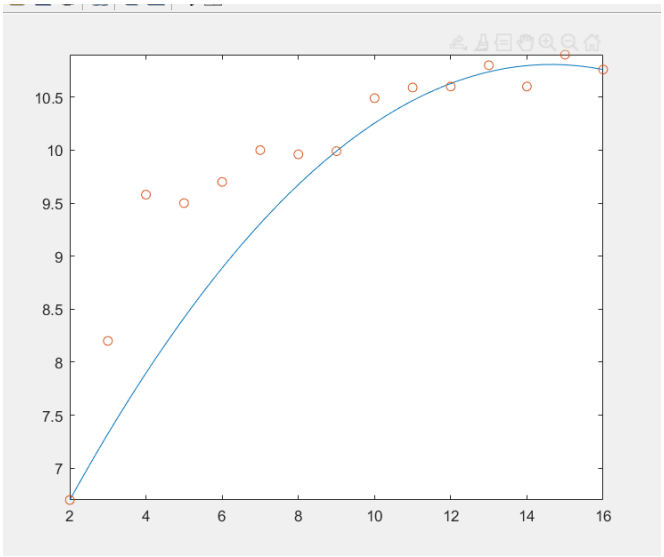
```
function [f] = Lagrange_1(x0,x1,y0,y1)%返回拉格朗日一次插值函数
    f = @(x)(y0 + (y1 - y0)/(x1 - x0)*(x - x0));
end
```

将起始点和最终点代入一次拉格朗日插值，可以得到一条一次函数曲线

```
function [f] = Lagrange_2(x0,x1,x2,y0,y1,y2)
    %返回拉格朗日二次插值函数
    f = @(x)( y0*(x - x1)*(x - x2)/(x0 - x1)/(x0 - x2) + y1*(x - x0)*(x - x2)/(x1 - x2)/(x1 - x0) + y2*(x - x0)*(x - x1)/(x2 - x1)/(x2 - x0));
end
```

代入起始点，中间点和最终点进行二次拉格朗日插值，可得到曲线如右下图





从图中可以看到，插值效果很不好，有很多点不在曲线上，此方法不适合该问题

拉格朗日高次插值计算无承袭性，每增加一个节点，所有基函数都要重新计算，且随着节点的增加，不仅高次 L(x)计算量大，精度也不一定好。

二、牛顿高次插值

牛顿低次插值方法与拉格朗日低次插值结果相似，这里就不做讨论了

牛顿插值法具有承袭性：$N_{n+1}(x) = N_n(x) + q_{n+1}(x)$

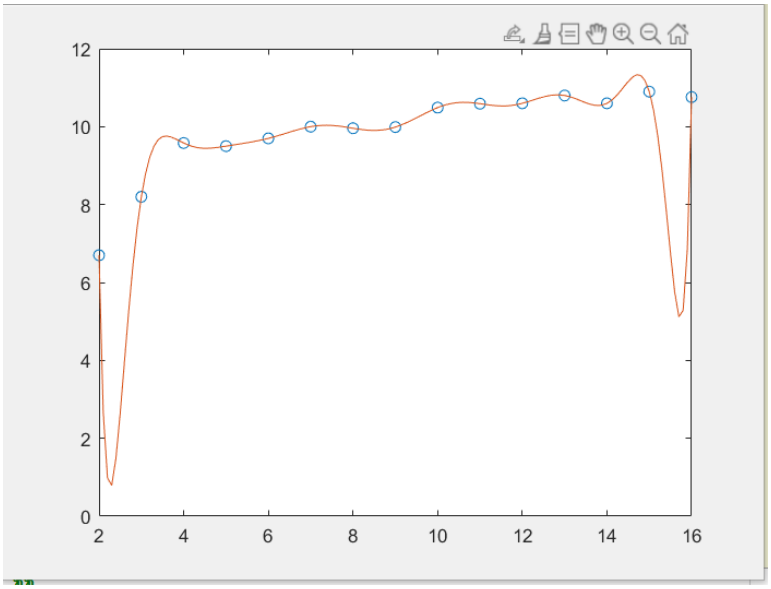
其中$N_n(x) = f(x_0) + f[x_0, x_1](x - x_0) + \cdots + f[x_0 \cdots x_n](x - x_0) \ldots (x - x_n)$

程序设计：先计算各阶差商，从而确定插值函数的系数，然后代入公式得到插值函数，进行插值函数图像的绘制

```
function  Newton(M)
    %构建保存差商斜边的向量a
    a = M(:,2);
    [n,~] = size(M);
    for i = 2:n
        deltx = [];
        delty = [];
        for k = i:n
            deltx(end+1) = M(k,1) - M(k-i+1,1); %计算差商的分子
            delty(end+1) = a(k) - a(k-1);       %计算差商的分母
        end
        for k = i:n
            a(k) = delty(k-i+1)/deltx(k-i+1);
        end
    end
```

```
    %构建牛顿插值函数
    f = @(x)(a(1));
    for i = 2:n
        f1 = @(x)(x-M(1,1))*a(i);
        for j = 2:i-1
            f1 = @(x)(f1(x)*(x - M(j,1)));
        end
        f = @(x)f(x)+f1(x);
    end

    %绘制函数
    X = [];
    Y = [];
    for i = M(1,1):0.1:M(n,1)
        X(end+1) = i;
        Y(end+1) = f(i);
    end

    plot(M(:,1),M(:,2),'o',X,Y);
end
```

程序运行后产生图像如下:



从图中可以看到，在中间段插值情况还可以，但在两端区间上发生了激烈振荡，插值多项式截断误差/计算余项偏大，出现了龙格现象，此现象说明了加密节点并不能保证所得到的插值多项式能够更好的逼近 f(x)，高次插值效果不一定比低次插值好。

三、分段低次插值

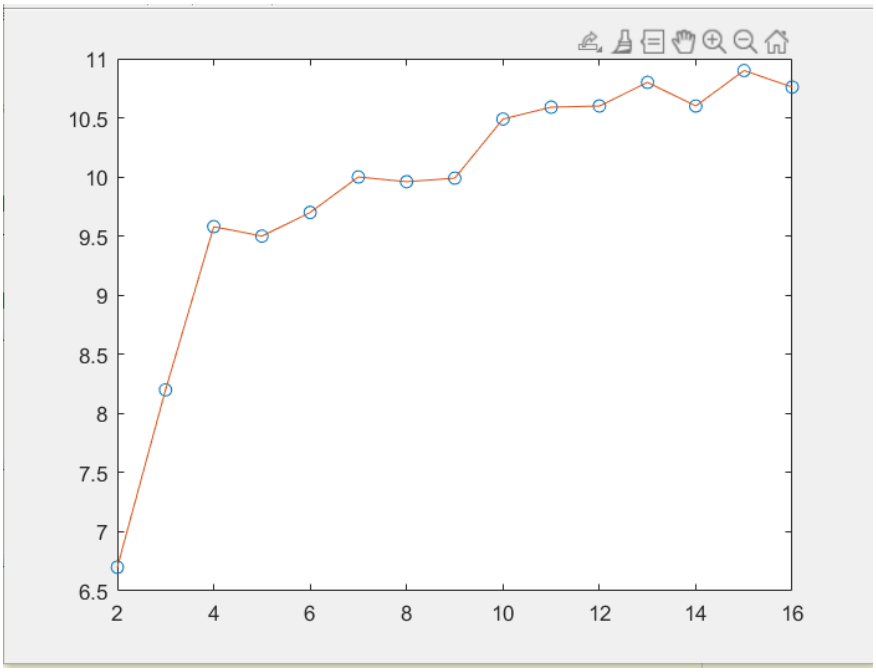
将整个插值区间分成若干个小区间，在每个小区间上，进行低次插值，当插值点 x 在第 i 各小区间上时，采用一次线性插值公式或二次插值公式进行小区插值。

分段一次插值:

在 15 组数据间，相邻两点之间进行一次拉格朗日插值运算，再绘制该区间内的函数曲线图象，代码如下:

```
function Interpolation_Split1(M)
    X = [];
    Y = [];
    [n,~] = size(M);

    for i = 1:n-1%n个点可以分成n-1个段，进行一次插值
        f = Lagrange_1(M(i,1),M(i+1,1),M(i,2),M(i+1,2));
        for x = M(i,1):0.1:M(i+1,1)
            X(end+1) = x;
            Y(end+1) = f(x);
        end
    end
    plot(M(:,1),M(:,2),'o',X,Y);

end
```

最后产生图像如下:

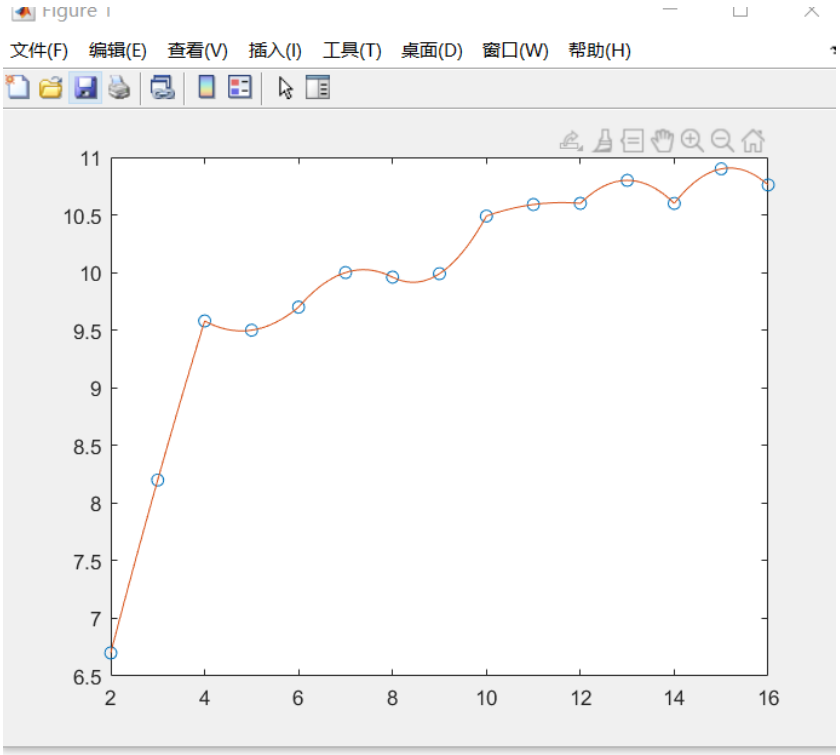


从上图可以看出：分段一次插值已经可以较好的反映曲线趋势

分段三次插值：
在 15 组数据间，相邻三点之间进行一次二次拉格朗日插值运算，再绘制该区间内的函数曲线图象，代码如下：

```
function Interpolation_Split2(M)
    X = [];
    Y = [];
    [n,~] = size(M);

    for i = 1:2:n-1%每次插值包含3组点数据，进行二次插值
        f = Lagrange_2(M(i,1),M(i+1,1),M(i+2,1),M(i,2),M(i+1,2),M(i+2,2));
        for x = M(i,1):0.1:M(i+2,1)
            X(end+1) = x;
            Y(end+1) = f(x);
        end
    end
    plot(M(:,1),M(:,2),'o',X,Y);

end
```

运行代码后，可得到如下图像：



可以看到分段二次插值图像优于分段一次插值图像，较为适合该方法。

分段插值的缺陷：不能保证节点处插值函数的导数连续，因而不能满足某些工程技术上曲线光滑性的要求。

四、样条插值

设 f(x)为区间[a,b]山的一个（m-1 次）连续可微函数。$a = x_0 < \cdots < x_n = b$

设函数：$S(x) = \begin{cases} S_1(x), x \in [x_0, x_1] \\ S_i(x), x \in [x_{i-1}, x_i] \\ S_n(x) \quad , \quad x \in [x_{n-1}, x_n] \end{cases}$

满足条件：1.S(x)在区间[a,b]上存在 m-1 阶连续导数

2.每个子区间$[x_{i-1}, x_i]$上$S_i(x)$都是一个不高于 m 次的多项式

3.满足插值条件$S_i(x) = f_i(x)$，称 S(x)为函数 f(x)关于节点的 m 次样条插值函数

二次样条插值：

$f(x) = a_i x_+^2 b_i x + c_i$ ，$x \in [x_{i-1}, x_i]$,i=1,2,3,⋯,n

满足条件：1.相邻多项式在内部节点处函数值相等

2.第一和最后一个函数通过端点

3.内部节点的一阶导数相等

4.第一个节点处的二阶导数为 0，即 a1=0，则连接前两个点的为一条直线

通过上述条件进行算法设计：

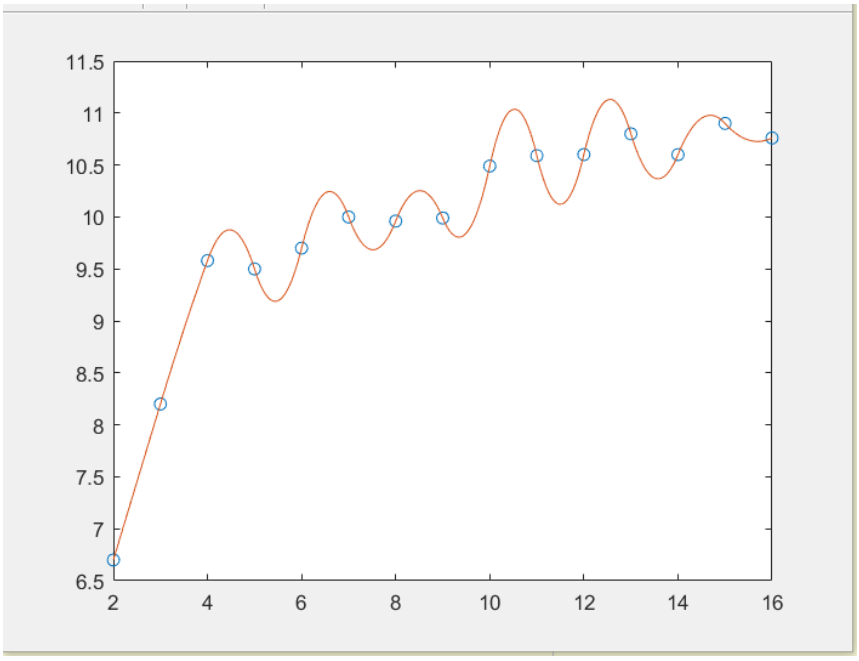
此方法需要解出 3n-4 个未知参数，而上述条件可列出 3n-4 个方程，即可把未知参数求解，最后回代入二次样条插值公式，即可获得插值图像。

Matlab 代码：

```
function Interpolation_Spine2(M)
    %参数矩阵计算
    [n,~] = size(M);
    A = zeros(3*n-4,3*n-4);%A为求取参数的系数矩阵
    b = zeros(3*n-4,1);%b为方程的右边值

    for i = 1:n-1
        if i == 1
            A(1:3,1:4)=[M(1,1),1,0,0;M(2,1),1,0,0;1,0,-2*M(2,1),-1];
            b(1:2) = [M(1,2),M(2,2)];
        elseif i ~= n-1
            A(3*i-2:3*i,3*i-3:3*i-1)=[M(i,1)^2,M(i,1),1;M(i+1,1)^2,M(i+1,1),1;2*M(i+1,1),1,0];
            A(3*i,3*i:3*i+2) = [-2*M(i+1,1),-1,0];
            b(3*i-2:3*i) = [M(i,2),M(i+1,2),0];
        else
            A(3*i-2:3*i-1,3*i-3:3*i-1) = [M(i,1)^2,M(i,1),1;M(i+1,1)^2,M(i+1,1),1];
            b(3*i-2:3*i-1) = [M(i,2),M(i+1,2)];
        end
    end
```

```
    S = Guass_Jordan(A, 3*n-4, b);
    %绘制二次样条图像
    X = [];
    Y = [];
    k = 1;
    for i = M(1:1):0.01:M(n, 1)
        if i == M(k+1, 1)
            k = k + 1;
        end
        X(end+1) = i;
        if k == 1
            Y(end+1) = S(1)*i+S(2);
        elseif k == n
            Y(end+1) = M(n, 2);
        else
            Y(end+1) = S(3*k-3)*(i^2)+S(3*k-2)*i+S(3*k-1);
        end
    end

    plot(M(:, 1), M(:, 2), 'o', X, Y);

end
```

运行代码，可获得下图：



可以看到二次样条插值图像具有很好的光滑性，能够很好的通过每个点，但在点与点之间的起伏较大。

三次样条插值：

$S_i(x) = a_i x^3 + b_i x^2 + ci^x + d_i$， $x \in [x_{i-1}, x_i]$,i=1,2,3,⋯,n

式中存在 4n 个未知数，满足如下条件：

1.端点值相等，i=1,2,3,⋯,n-1

2.端点的一阶导值相等，i=1,2,3,⋯,n-1

3.端点的二阶导值相等，i=1,2,3,⋯,n-1

4. $S_i(x) = f_i(x)$，i=0,2,3,⋯,n

以上共 4n-2 个式子，无法满足条件，所以需要边界条件的限制，本例中假设了在起始点和最终点的二阶导数已知且都为 0 的自然边界条件。

算法设计：

利用三弯矩法实现三次样条插值

利用 S(x)在节点 x_i 处的二阶导数值 $M_i=S''(x_i)$表示 $S_i(x)$。用 S'(x)在内节点 x_i 上的连续性和边界条件来确定 M_i，利用二阶导数连续性和插值条件端点值相等和二阶导数相等

得到此式：$u_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = g_i$　i=1,2,3,⋯,n-1

其中：$u_i = \frac{h_i}{h_i + h_{i+1}}$　$\lambda_i = 1 - u_i$　$g_i = 6f[x_{i-1}, x_i, x_{i+1}]$　$h_i = x_i - x_{i-1}$

再根据自然边界条件可得到如下矩阵：

A = $\begin{pmatrix} 2 & \lambda 1 & & \cdots & 0 & 0 & 0 \\ \mu 2 & 2 & \lambda 2 & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & u_{n-2} & 2 & \lambda_{n-2} \\ 0 & 0 & 0 & \cdots & u_{n-1} & & 2 \end{pmatrix}$

X = [M1 ,⋯,M_{n-1}]

B = [g1，g2，⋯,g_{n-1}]

求解出 X，将参数代入到三次样条插值函数公式

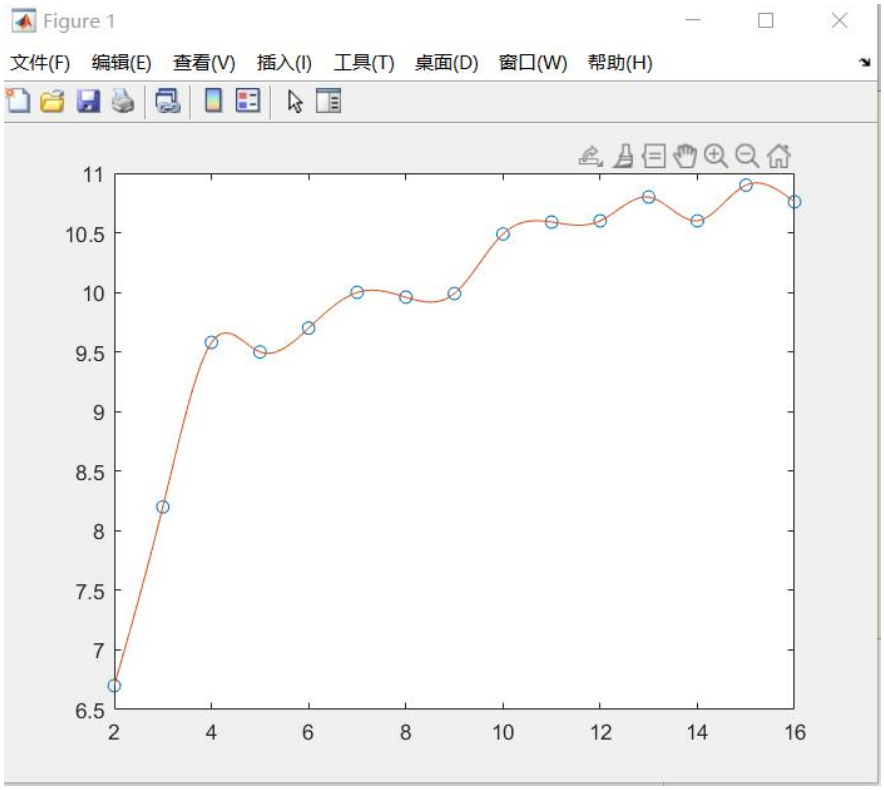
$$S_i(x) = \frac{1}{6h_i}[M_{i-1}(x_i - x)^3 + M_i(x - x_{i-1})^3] + (f_{i-1} - \frac{M_{i-1}h_i^2}{6})\frac{x_i - x}{h_i} + (f_i - \frac{M_i h_i^2}{6})\frac{x - x_{i-1}}{h_i}$$

绘制图像即可。

Matlab 代码:

```
function Interpolation_Spine3(M)
    %假设满足x0处的二阶导和xn处的二阶导均为0，自然边界条件
    h = [];
    [n,~] = size(M);
    for i = 2:n
        h(end+1) = M(i,1) - M(i-1,1);
    end
    lamuda = [];
    miu = [];%lamuda和miu为记录值
    g = [];%g为方程右边值
    for i = 1:n-2
        lamuda(end+1) = h(i+1)/(h(i) + h(i+1));
        miu(end+1) = h(i)/(h(i) + h(i+1));
        g(end+1) = 6*(((M(i+2,2)-M(i+1,2))/(M(i+2,1) - M(i+1,1)) - (M(i+1,2)-M(i,2))/(M(i+1,1) - M(i,1)))/(h(i)+h(i+1)));
    end

    A = diag(diag(2*ones(n-2)));

    A(1,2) = lamuda(1);

%构建系数矩阵

for i=2:n-3
   A(i,i-1) = miu(i);
   A(i,i+1) = lamuda(i);
end
A(n-2,n-3) = miu(n-2);

%求解一阶导数向量M1
M1 = Guass_Jordan(A,n-2,g');
M1 = [0;M1;0];
%三次样条函数
X = [];
Y = [];
k = 1;
for i = M(1:1):0.1:M(n,1)
    if i == M(k+1,1)
        k = k + 1;
    end
    X(end+1) = i;
    if k~= n
        Y(end+1) = 1/(6*h(k))*(M1(k)*((M(k+1,1)-i)^3) + M1(k+1)*((i-M(k,1))^3)) + (M(k,2)-M1(k)*(h(k)^2)/6)*(M(k+1,1)-i)/h(k) + (M(k+1,2) - M1(k+1)*(h(k)^2)/6)*(i-M(k,1))/h(k);
    else
        Y(end+1) = M(n,2);
    end
end


    %绘制图像
    plot(M(:,1),M(:,2),'o',X,Y)

end
```

程序运行结果如下:



从图中可以看到，此方法得到的插值图像最为平滑，插值效果最好。

样条插值和分段插值的比较：

1.简单插值的次数与节点个数有关，n+1 个节点上要用 n 次多项式来插值，而样条插值多项的次数与节点个数无关，便于在多个节点上用低次插值

2. 一般的分段低次插值，每段上插值多项式不同，各段表达式之间没有内在联系，而样条插值多项式各段之间有联系。

3. 样条多项式往往只需已知 m-1 个边界节点上的导数值就够了，其余节点上的导数值是自然形成的。

插值方法的缺陷：

1. 由实验提供的数据带有测试误差，插值函数会保留数据的全部测量误差

2. 当插值函数的阶数较高时，曲线摆动很大，而求得的插值函数与实验规律可能偏离甚远

3. 实验数据往往很多，用插值法得到的近似表达式缺乏实用价值

五、拟合

构造一个能逼近列表数据的近似的数学表达式，使各数据点从总体上最贴近，而不一定要求构造的函数曲线通过所给数据点

最小二乘法：线性拟合方法即$\varphi(x) = a + bx$

建立此方程组：$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix}$ 求解系数 a 和 b

程序为 fit.m

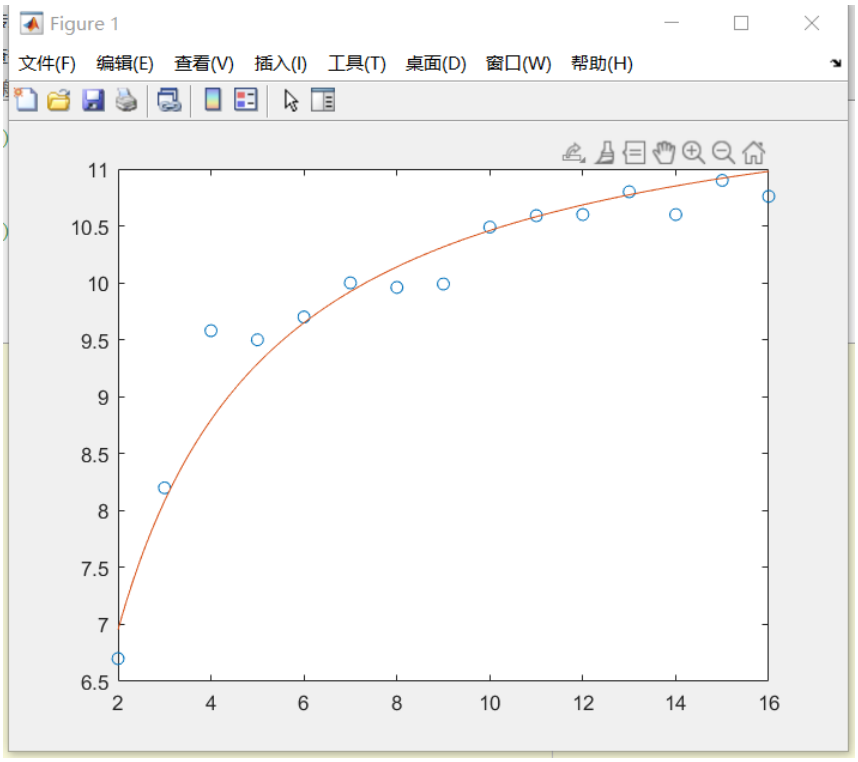
如下：

```
function [a,b] = fit(x,y)  %最小二乘法拟合曲线,传入的x和y为两个行向量
    %求解最小二乘法的系数a和b
    %构建系数矩阵和方程右边列向量
    A = zeros(2,2);
    [~,n] = size(x);
    A(1,1) = n;
    A(1,2) = sum(x);
    A(2,1) = A(1,2);
    B = zeros(2,1);
    B(1) = sum(y);
    for i = 1:n
        A(2,2) = A(2,2) + x(i)^2;
        B(2) = B(2) + x(i)*y(i);
    end

    X = Guass_Jordan(A,2,B);%高斯约旦法求解出一个[a b]的2×1的列向量
    a = X(1);
    b = X(2);


end
```

拟合方法 1：

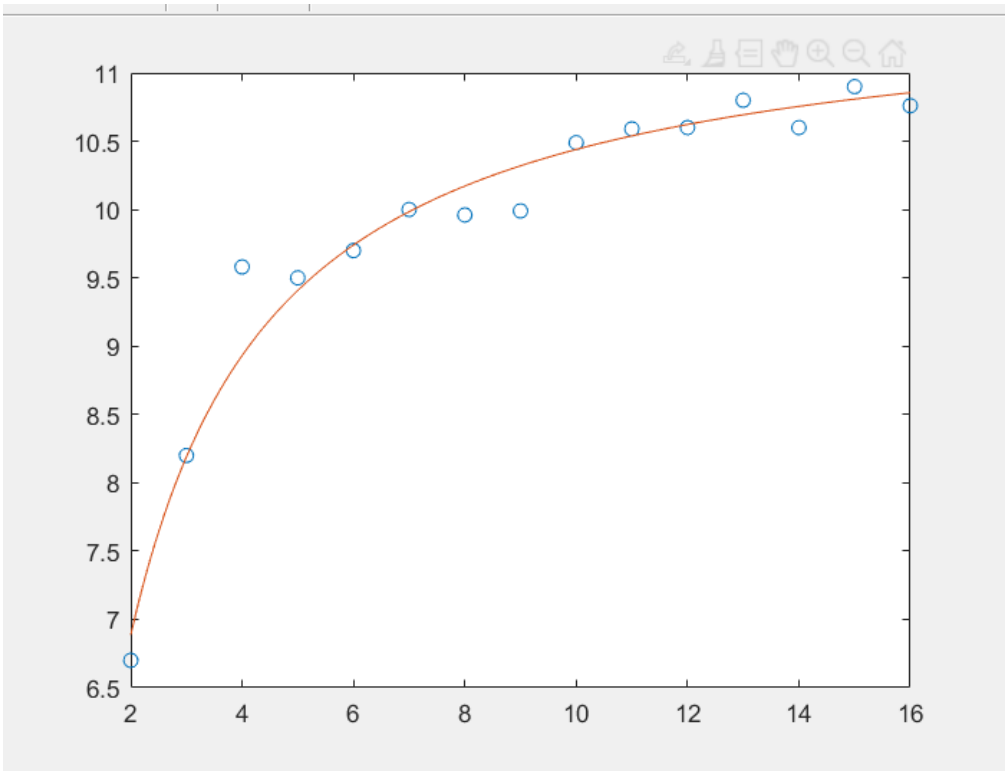
双曲线型函数：$\frac{1}{y} = a + \frac{b}{x}$，传入$\frac{1}{y}$和$\frac{1}{x}$的列表进入 fit 函数，得到系数 a 和 b 代入，即为所需的双曲线型函数

通过 myplot 进行画图和均方误差计算，结果如下：



此方法的均方误差为0.066884>>

2.指数型函数拟合：$y = ae^{\frac{b}{x}}$，将 lny 和$\frac{1}{x}$的列表进入 fit 函数，得到系数 a 和 b，将 lna 和 b 代入，即为所需的双曲线型函数



```
此方法的均方误差为0.045192>>
```

比较两方法的均方误差，可以发现第二种方法的均方误差更小，第二种方法拟合效果更好。

下附：程序中写到的 myplot 代码：

```
文件  导航  编辑  断点
function myplot(f,x,y)
    %绘制图像
    X = [];
    Y = [];
    for i = x(1):0.1:x(end)
        X(end+1) = i;
        Y(end+1) = f(i);
    end

    plot(x,y,'o',X,Y);

    %均方误差计算
    sum = 0;
    for i = 1:1:length(x)
        sum = sum + (y(i) - f(x(i)))^2;
    end
    fprintf("此方法的均方误差为" + sum/length(x));

end
```

体会：经过这次作业，我了解到了多种插值和拟合的方法，以及他们所适用的条件。了解了通过测得的数据点获得一条近似的曲线的方法，也锻炼了我的代码能力，积累了我的函数库，相信后面的大作业写起来也将是如鱼得水。