# 曲线拟合的数值计算方法实验

姓名：王旋宇
学号：2012049010022
班级：数理基科班
老师：赖生建

2014-12-17

# 1 实验目的

1. 了解最小二乘法的原理，会使用最小二乘法计算最小二乘拟合曲线，并编程实现。
2. 了解线性最小二乘法的原理，会使用线性最小二乘法计算线性独立函数的线性组合而成的函数，并编程实现
3. 了解三次样条函数的性质。了解五种三次样条端点约束的内容。会编程实现三次紧压样条约束求解样条系数。
4. 了解三角多项式逼近，会计算任意周期函数的离散傅里叶级数，并编程实现。
5. 了解贝塞尔曲线，并会根据给出控制点集写出N阶贝塞尔曲线的表达式。

# 2 实验原理

## 2 .1 最小二乘拟合曲线

**定义1 .** 误差及其多种表现形式

$$\begin{aligned}
\text{误差（又称偏差或残差）} &: e_k = f(x_k) - y_k \quad \text{其中} 1 \le k \le N && (1)\\
\text{最大误差} &: E_\propto(f) = \max_{1\le k\le N}\{|f(x_k) - y_k|\} && (2)\\
\text{平均误差} &: E_1(f) = \frac{1}{N}\sum_{k=1}^{N}|f(x_k) - y_k| && (3)\\
\text{均方根误差} &: E_2(f) = (\frac{1}{N}\sum_{k=1}^{N}|f(x_k) - y_k|^2)^{1/2} && (4)
\end{aligned}$$

**定义2 .** 最小二乘曲线$y = f(x) = Ax + B$是满足均方根误差$E_2(f)$ 最小的曲线

**定理1 .** 设$\{(x_k, y_k)\}_{k=1}^{N}$有$N$个点，其中横坐标$x_k{}_{k=1}^{N}$ 是确定的。最小二乘拟合曲线$y = Ax + B$的系数就是下列线性方程组的解，这些方程成为**正规方程**。

$$\begin{aligned}
(\sum_{k=1}^{N}x_k^2)A + (\sum_{k=1}^{N}x_k)B &= (\sum_{k=1}^{N}x_k y_k)\\
(\sum_{k=1}^{N}x_k)A + NB &= (\sum_{k=1}^{N}y_k)
\end{aligned} \tag{5}$$

## 2 .2　线性最小二乘法

设有N个数据点$\{(x_k, y_k)\}$，并给定M 个线性独立函数$\{f_i(x)\}$。为求M个系数$\{c_j\}$，使用由线性组合形成的函数$f(x)$，表示为

$$f(x) = \sum_{j=1}^{M} c_j f_j(x) \tag{6}$$

求解最小误差平方和，表示为

$$E(c_1, c_2, \cdots, c_m) = \sum_{k=1}^{N} (f(x_k) - y_k)^2 = \sum_{k=1}^{N} \left( \left( \sum_{j=1}^{M} c_j f_j(x_k) \right) - y_k \right)^2 \tag{7}$$

为求解E的最小值，每个偏导数必须为零，这样可以得到如下方程组

$$\sum_{k=1}^{N} \left( \left( \sum_{j=1}^{M} c_j f_j(x_k) \right) - y_k \right) (f_i(x_k)) = 0 \quad \text{其中} i = 1, 2, \cdots, M \tag{8}$$

交换方程组中(**??**)中的仇和顺序，可得一个$M \times M$线性方程组，未知数是系数$\{c_j\}$。该方程组称为正规方程：

$$\sum_{j=1}^{M} \left( \sum_{k=1}^{N} f_i(x_k) f_j(x_k) \right) c_j = \sum_{k=1}^{N} f_i(x) y_k \quad \text{其中} i = 1, 2, \cdots, M \tag{9}$$

## 2 .3　三次紧压样条

**定义3 .** 设$\{(x_k, y_k)\}_{k=0}^{N}$有*N+1* 个点，其中$a = x_0 < x_1 < \cdots < x_N = b$。如果存在$N$ 个三次多项式$S_k(x)$，系数为$s_{k,0}, s_{k,1}, s_{k,2}, s_{k,3}$，满足如下性质：

$$\begin{aligned}
&I. \quad S(x) = S_k(x) = s_{k,0} + s_{k,1}(x - x_k) + s_{k,2}(x - x_k)^2 + s_{k,3}(x - x_k)^3 \\
&\qquad\qquad x \in [x_k, x_{k+1}], \quad k = 0, 1, \cdots, N-1 \\
&II. \quad S(x_k) = y_k \qquad k = 0, 1, \cdots, N \\
&III. \quad S_k(x_k + 1) = S_{k+1}(x_k + 1) \quad k = 0, 1, \cdots, N-2 \\
&IV. \quad S'_k(x_k + 1) = S'_{k+1}(x_k + 1) \quad k = 0, 1, \cdots, N-2 \\
&V. \quad S''_k(x_k + 1) = S''_{k+1}(x_k + 1) \quad k = 0, 1, \cdots, N-2
\end{aligned}$$

则称函数$S(x)$为三次样条函数。

性质II.到V.提供了4N-2个条件。这样剩下两个自由度，可称之为端点约束：设计在点$x_0$和$x_N$处的导数$S'(x_0)$和$S''(x)$。

**引理2 .** 存在唯一的三次样条曲线，其一阶导数的边界条件是$S'(a) = d_0$和$S'(b) = d_N$

根据引理(**??**)可以得出三件线性方程组$HM = V$表示为

$$H = \begin{bmatrix} \frac{2}{3}(h_0+2h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & & \ddots & & & \\ & & & h_{N-3} & 2(h_{N-3}+h_{N-2}) & h_{N-2} \\ & & & & h_{N-2} & 2h_{N-2}+\frac{2}{3}h_{N-1} \end{bmatrix}$$

$$M = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_{N-2} \\ m_{N-1} \end{bmatrix} \quad V = \begin{bmatrix} u_1 - 3(d_0 - S'(x_0)) \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} - 3(S'(x_0) - d_{N-1}) \end{bmatrix} \tag{10}$$

其中$h_k = x_{k+1} - x_k, d_k = \frac{y_{k+1}-y_k}{h_k}, u_k = 6(d_k - d_{k-1})$ 然后由

$$m_0 = \frac{3}{h_0}(d_0 - S'(x_0)) - \frac{m_1}{2} \tag{11}$$

$$m_N = \frac{3}{h_{N-1}}(S'(x_N) - d_{N-1}) - \frac{m_{N-1}}{2} \tag{12}$$

$$s_{k,0} = y_k \tag{13}$$

$$s_{k,1} = d_k - \frac{h_k(2m_k + m_{k+1})}{6} \tag{14}$$

$$s_{k,2} = \frac{m_k}{2} \tag{15}$$

$$s_{k,3} = \frac{m_{k+1} - m_k}{6h_k} \tag{16}$$

可以得出所有的样条系数$\{s_{k,j}\}$

## 2 .4　三角多项式逼近

**定义4 .** 具有如下形式的级数：

$$T_M(x) = \frac{a_0}{2} + \sum_{j=1}^{M}(a_j cos(jx) + b_j sin(jx)) \tag{17}$$

称为*M*阶三角多项式

**定理3 .** 设有*N+1*个点$\{x_j, y_j\}_{j=0}^{N}$，其中$y_j = f(x_j)$，而且横坐标之间等距，即：

$$x_j = -\pi + \frac{2j\pi}{N} \quad \text{其中} j = 0, 1, \cdots, N \tag{18}$$

如果$f(x)$的周期为$2\pi$,而且$2M < N$，则存在式(**??**) 所示的三角多项式$T_M(x)$，使得下式的值最小。

$$\sum_{k=1}^{N}(f(x_k) - T_M(x_k))^2 \tag{19}$$

多项式系数$a_j$和$b_j$可通过如下公式计算

$$a_j = \frac{2}{N}\sum_{k=1}^{N} f(x_k)cos(jx_k) \quad \text{其中} j = 0, 1, \cdots, M \tag{20}$$

和

$$b_j = \frac{2}{N}\sum_{k=1}^{N} f(x_k)sin(jx_k) \quad \text{其中} j = 1, 2, \cdots, M \tag{21}$$

### 2 .5 贝塞尔曲线

**定义5 .** $N$阶贝恩斯坦多项式定义为

$$B_{i,N}(t) = \begin{pmatrix} N \\ i \end{pmatrix} t^i(1-t)^{N-i} i = 0, 1, 2, \cdots, \text{其中} \begin{pmatrix} N \\ i \end{pmatrix} = \frac{N!}{i!(N-i)!} \tag{22}$$

**定义6 .** 给定一个控制点集，$\{P_i\}_{i=0}^{N}$，其中$P_i = (x_i, y_i)$，定义

$$P(t) = \sum_{i=0}^{N} P_i B_{i,N}(t) \tag{23}$$

为$N$阶贝塞尔曲线，其中$B_{i,N}(t), i = 0, 1, \cdots, N$ 是$N$阶伯恩斯坦多项式，$t \in [0, 1]$

## 3 实验内容

### 3 .1 P202 1

胡克(Hooke)定律指出$F = kx$，其中F是拉伸弹簧的拉力，单位为盎司，x是拉伸长度，单位为英寸。根据下列实验数据，求解拉伸常量k的近似值。

| $x_k$ | $F_k$ |
|---|---|
| 0.2 | 3.6 |
| 0.4 | 7.3 |
| 0.6 | 10.9 |
| 0.8 | 14.5 |
| 1.0 | 18.2 |

| $x_k$ | $F_k$ |
|---|---|
| 0.2 | 5.3 |
| 0.4 | 10.6 |
| 0.6 | 15.9 |
| 0.8 | 21.2 |
| 1.0 | 26.4 |

## 3 .2 P215 1

洛杉矶郊区在11月8日的温度记录如下所示。共有24 个数据点。

**(a)** 根据例5.5中的处理过程，对给定的数据求解最小二乘曲线$f(x) = Acos(Bx)+ Csin(Dx) + E$

**(b)** 求$E_2(f)$

**(c)** 在同一坐标系中画出这些点集和(a) 得出的最小二乘曲线。

| 时间,p.m. | 温度 | 时间,a.m. | 温度 |
|---|---|---|---|
| 1 | 66 | 1 | 58 |
| 2 | 66 | 2 | 58 |
| 3 | 65 | 3 | 58 |
| 4 | 64 | 4 | 58 |
| 5 | 63 | 5 | 57 |
| 6 | 63 | 6 | 57 |
| 7 | 62 | 7 | 57 |
| 8 | 61 | 8 | 58 |
| 9 | 60 | 9 | 60 |
| 10 | 60 | 10 | 64 |
| 11 | 59 | 11 | 67 |
| 12 | 58 | 12 | 68 |

## 3 .3 P229 1

一个轿车在时间$t_k$时经过的距离为$d_k$，如下表所示。根据一阶导数边界条件$S'(0) = 0$ 和$S'(8) = 98$，求这些数据的三次紧压样条插值。

| 时间，$t_k$ | 0 | 2 | 4 | 6 | 8 |
|---|---|---|---|---|---|
| 距离，$d_k$ | 0 | 40 | 160 | 300 | 480 |

## 3 .4 P238 5

洛杉矶郊区在11月8日的温度记录如下所示。共有24 个数据点。

**(a)** 求三角多项式$T_7(x)$

**(b)** 在同一坐标系下画出图$T_7(x)$ 和24 个数据点。

**(c)** 使用本地的温度重新求解问题(a) 和问题(b)。

| 时间,p.m. | 温度 | 时间,a.m. | 温度 |
|---|---|---|---|
| 1 | 66 | 1 | 58 |
| 2 | 66 | 2 | 58 |
| 3 | 65 | 3 | 58 |
| 4 | 64 | 4 | 58 |
| 5 | 63 | 5 | 57 |
| 6 | 63 | 6 | 57 |
| 7 | 62 | 7 | 57 |
| 8 | 61 | 8 | 58 |
| 9 | 60 | 9 | 60 |
| 10 | 60 | 10 | 64 |
| 11 | 59 | 11 | 67 |
| 12 | 58 | 12 | 68 |

### 3 .5 P246 3

编写Matlab程序，生成并绘制组合贝塞尔曲线。利用该程序生成和绘制过3个控制点集$\{(0,0),(1,2),(1,1),(3,0)\}$,$\{(3,0),(4,-1),(5,-2),(6,1),(7,0)\}$,$\{(7,0),(4,-3),(2,-1),(0,0)\}$的贝塞尔曲线。

## 4 实验分析

### 4 .1 P202 1

应用最小二乘法求解$F = Ax + B$，其中A就是待求解的拉伸常量的近似值

### 4 .2 P215 1

待求解曲线是

$$f(x) = Acos(Bx) + Csin(Dx) + E \tag{24}$$

由于该方程是非线性的，并且很难线性化，使用牛顿非线性迭代法的话也很难确定一个较好的初始值，所以把题目简化一下，把B、D 看作是常数。把数据输入Matlab 中，利用Curve Fitting 工具对$f(x) = Acos(Bx) + Csin(Dx) + E$进行拟合，拟合采用非线性最小二乘法，算法为Trust-Region 算法。拟合得到B=3.267,D=-0.606. 所以

$$E(A,C,E) = \sum_{k=1}^{N}(Acos(Bx_k) + Csin(Dx_k) + E - y_k)^2 \tag{25}$$

令偏导数$\partial E/\partial A, \partial E/\partial C, \partial E/\partial E$ 为零，可得

$$\begin{aligned}
0 = \frac{\partial E(A,C,E)}{\partial A} &= 2\sum_{k=1}^{N}(Acos(Bx_k) + Csin(Dx_k) + E - y_k)cos(Bx_k) \\
0 = \frac{\partial E(A,C,E)}{\partial C} &= 2\sum_{k=1}^{N}(Acos(Bx_k) + Csin(Dx_k) + E - y_k)sin(Dx_k) \\
0 = \frac{\partial E(A,C,E)}{\partial A} &= 2\sum_{k=1}^{N}(Acos(Bx_k) + Csin(Dx_k) + E - y_k)
\end{aligned} \tag{26}$$

式(??)就是求解A，C，E的线性方程组。

## 4 .3 P229 1

根据(??),根据题中给出的数据算出矩阵中各元素的值。然后解矩阵。再根据(??)算出所需要的样条系数。

## 4 .4 P238 5

首先根据式(??)把自变量区间[0,24] 映射到区间$[\pi, \pi]$ 中：

$$\begin{aligned}
\phi: \quad & x \mapsto x' \\
& x' = \tfrac{(x-12)\pi}{12}
\end{aligned}$$

使得自变量符合计算离散傅里叶级数的条件，然后根据(??) 和(??) 来计算三角多项式的系数$\{a'_j\}_{j=0}^{M}, \{b'_j\}_{j=0}^{M}$。得到$T'_M(x)$。然后将自变量逆映射到原来的区间，得到$T_M(x) = T'_M(x\pi/12)$

## 4 .5 P246 3

根据N阶贝塞尔曲线的定义，由给出的三组控制点集$\{(0,0),(1,2),(1,1),(3,0)\}$，$\{(3,0),(4,-1),(5,-2),(6,1),(7,0)\}$，$\{(7,0),(4,-3),(2,-1),(0,0)\}$，我们可以得出三个控制曲线的表达式：

$$\begin{cases}
x_1(t) = 0B_{0,3}(t) + 1B_{1,3}(t) + 1B_{2,3}(t) + 3B_{3,3}(t) \\
y_1(t) = 0B_{0,3}(t) + 2B_{1,3}(t) + 1B_{2,3}(t) + 0B_{3,3}(t) \\
x_2(t) = 3B_{0,4}(t) + 4B_{1,4}(t) + 5B_{2,4}(t) + 6B_{3,4}(t) + 7B_{4,4}(t) \\
y_2(t) = 0B_{0,4}(t) - 1B_{1,4}(t) - 2B_{2,4}(t) + 1B_{3,4}(t) + 0B_{4,4}(t) \\
x_3(t) = 7B_{0,3}(t) + 4B_{1,3}(t) + 2B_{2,3}(t) + 0B_{3,3}(t) \\
y_3(t) = 0B_{0,3}(t) - 3B_{1,3}(t) - 1B_{2,3}(t) + 0B_{3,3}(t)
\end{cases} \tag{27}$$

带入公式(**??**)中的伯恩斯坦多项式得到

$$\begin{cases} x_1(t) =0(1-t)^2+1\cdot 3t(1-t)^2+1\cdot 3t^2(1-t)+3\cdot t^3 \\ y_1(t) =0(1-t)^2+2\cdot 3t(1-t)^2+1\cdot 3t^2(1-t)+0\cdot t^3 \\ x_2(t) =3\cdot(1-4t+6t^2-4t^3+t^4)+4\cdot(4t-12t^2+12t^3-4t^4) \\ \qquad +5\cdot(6t^2-12t^3+6t^4)+6\cdot(4t^3-4t^4)+7\cdot t^4 \\ y_2(t) =0\cdot(1-4t+6t^2-4t^3+t^4)-1\cdot(4t-12t^2+12t^3-4t^4) \\ \qquad -2\cdot(6t^2-12t^3+6t^4)+1\cdot(4t^3-4t^4)+0\cdot t^4 \\ x_3(t) =7(1-t)^2+4\cdot 3t(1-t)^2+2\cdot 3t^2(1-t)+0\cdot t^3 \\ y_3(t) =0(1-t)^2-3\cdot 3t(1-t)^2-1\cdot 3t^2(1-t)+0\cdot t^3 \end{cases} \tag{28}$$

然后使用Matlab画出曲线即可。

# 5 实验结果

## 5 .1 P202 1

对于第一组数据，实验结果如下所示。

| $x_k$ | $y_k$ | $x^2$ | $xy$ |
|---|---|---|---|
| 0.2 | 3.6 | 0.04 | 0.72 |
| 0.4 | 7.3 | 0.16 | 2.92 |
| 0.6 | 10.9 | 0.36 | 6.54 |
| 0.8 | 14.5 | 0.64 | 11.6 |
| 1.0 | 18.2 | 1.00 | 18.2 |

(a) 第一组数据的预处理

| | |
|---|---|
| $\sum_{k=1}^{N} x_k^2$ | 2.2 |
| $\sum_{k=1}^{N} x_k$ | 3 |
| $\sum_{k=1}^{N} x_k y_k$ | 39.98 |
| $\sum_{k=1}^{N} y_k$ | 54.5 |

(b) 第一组的矩阵系数

Table 1: 第一组的数据处理

所以得到结果：

| | A | B |
|---|---|---|
| 计算结果 | 18.2 | -0.02 |
| Matlab计算对照 | 18.2 | -0.02 |

Table 2: 第一组的计算结果

对于第二组数据，实验结果如下所示。

| $x_k$ | $y_k$ | $x^2$ | $xy$ |
|---|---|---|---|
| 0.2 | 5.3 | 0.04 | 1.06 |
| 0.4 | 10.6 | 0.16 | 4.24 |
| 0.6 | 15.9 | 0.36 | 9.54 |
| 0.8 | 21.2 | 0.64 | 16.96 |
| 1.0 | 26.4 | 1.00 | 26.4 |

(a) 第二组数据的预处理

| | |
|---|---|
| $\sum_{k=1}^{N} x_k^2$ | 2.2 |
| $\sum_{k=1}^{N} x_k$ | 3 |
| $\sum_{k=1}^{N} x_k y_k$ | 58.2 |
| $\sum_{k=1}^{N} y_k$ | 79.4 |

(b) 第二组的矩阵系数

Table 3: 第二组的数据处理

所以得到结果：

| | A | B |
|---|---|---|
| 计算结果 | 18.2 | -0.02 |
| Matlab计算对照 | 18.2 | -0.02 |

Table 4: 第二组的计算结果

## 5 .2 P215 1

| | |
|---|---|
| $\sum_{k=1}^{N} cos^2(Bx_k)$ | 11.474603 |
| $\sum_{k=1}^{N} cos(Bx_k)sin(Dx_k)$ | 0.554774 |
| $\sum_{k=1}^{N} cos(Bx_k)$ | -0.999789 |
| $\sum_{k=1}^{N} sin^2(Dx_k)$ | 12.683914 |
| $\sum_{k=1}^{N} sin(Dx_k)$ | -2.691411 |
| $\sum_{k=1}^{N} cos(Bx_k)y_k$ | -66.841075 |
| $\sum_{k=1}^{N} sin(Dx_k)y_k$ | -194.099505 |
| $\sum_{k=1}^{N} y_k$ | 1467.000000 |

(a) 矩阵参数

| | A | C | E |
|---|---|---|---|
| 计算结果 | -0.409146 | -2.374872 | 60.841633 |
| Matlab计算对比 | -0.4091 | -2.375 | 60.84 |

(b) 求得的解

Table 5: 线性最小二乘法计算结果

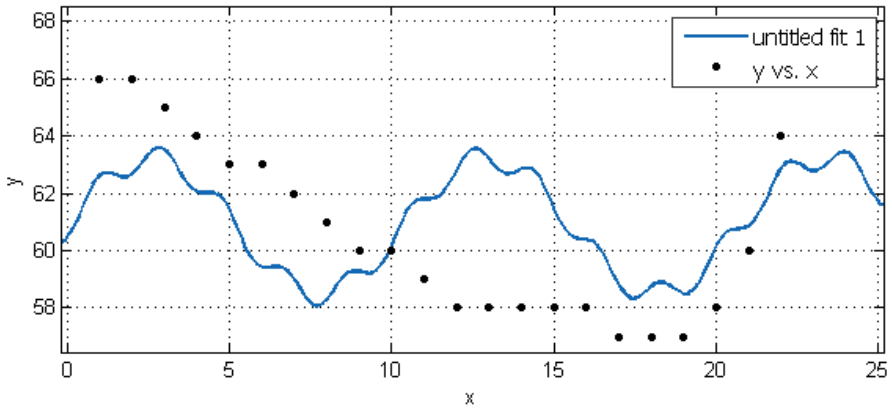



Fig 1: 线性最小二乘法拟合效果图

其残差平方和为214.

## 5 .3 P229 1

| | |
|---|---|
| $h_0$ | 2 |
| $h_1$ | 2 |
| $h_2$ | 2 |
| $h_3$ | 2 |

(a) h的值

| | |
|---|---|
| $d_0$ | 20 |
| $d_1$ | 60 |
| $d_2$ | 70 |
| $d_3$ | 90 |

(b) d的值

| | |
|---|---|
| $u_1$ | 240 |
| $u_2$ | 60 |
| $u_3$ | 120 |

(c) u的值

| | |
|---|---|
| $v_1$ | 180 |
| $v_2$ | 60 |
| $v_3$ | 96 |

(d) V的值

| | |
|---|---|
| $m_0$ | 16.75 |
| $m_1$ | 26.5 |
| $m_2$ | -2.75 |
| $m_3$ | 14.5 |
| $m_4$ | 4.75 |

(e) m的值

Table 6: 数据预处理

| | 常数项 | 一次项 | 二次项 | 三次项 |
|---|---|---|---|---|
| $[0, 2]$ | 0 | 0 | 8.375000 | 0.8125 |
| $[2, 4]$ | 40 | 49.083333 | 13.25 | -2.4375 |
| $[4, 6]$ | 160 | 52.666667 | -1.375 | 1.4375 |
| $[6, 8]$ | 300 | 75.25 | 7.25 | -0.8125 |

Table 7: 解得的样条系数

## 5 .4 P238 5

计算得到的三角多项式系数如下表所示

| | 自编程序计算结果 | 教材中Matlab计算结果 |
|---|---|---|
| $a_1$ | 60.913043 | 63.7826 |
| $a_2$ | -3.613490 | -3.6975 |
| $a_3$ | 1.678610 | 1.6428 |
| $a_4$ | -1.050496 | -1.0751 |
| $a_5$ | 0.286623 | 0.2174 |
| $a_6$ | -0.262210 | -0.0455 |
| $a_7$ | -0.085918 | -0.1739 |
| $a_8$ | 0.013043 | 0.0268 |

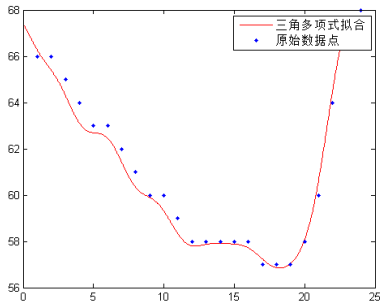
(a) $a_j$

| | 自编程序计算结果 | 教材中Matlab计算结果 |
|---|---|---|
| $b_1$ | -2.220780 | -2.7503 |
| $b_2$ | 0.265698 | 0.4868 |
| $b_3$ | 0.565878 | 0.3584 |
| $b_4$ | -0.432946 | -0.3765 |
| $b_5$ | 0.162230 | 0.0810 |
| $b_6$ | -0.134709 | -0.2609 |
| $b_7$ | 0.175486 | 0.1167 |

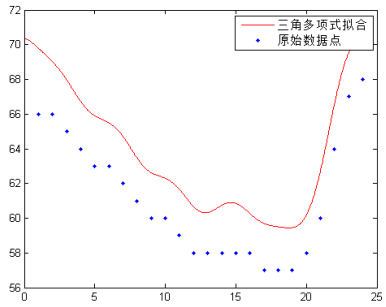
(b) $b_j$

Table 8: 三角多项式的系数

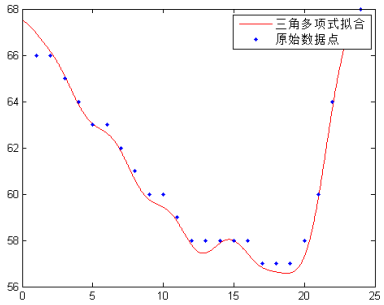
三角多项式的拟合效果如图




(a) 自己计算的拟合效果




(b) 教材Matlab程序拟合效果




(c) 教材中程序修改$a_0$之后

Fig 2: 三角多项式的拟合效果图

## 5 .5 P246 3

过三个控制点集的贝塞尔曲线如图所示。

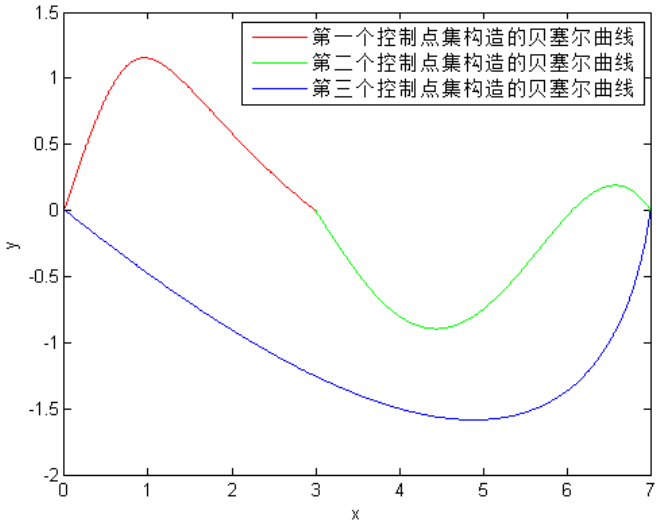



Fig 3: 过三个控制点集的贝塞尔曲线

# 6 实验结果分析

- 最小二乘法在待拟合方程为函数的线性组合的时候或者可以化为线性的时候最好用，其他的时候还是需找其他算法比较好。或者在实际应用中，尽量让变量之间的关系可以变为线性形式，这样更有利于找到变量之间的关系。

- 线性最小二乘法可以看图看出来拟合效果勉强可以，这个可能和简化了参数有关。但是可以看出来除了10到15之间的那一段之外整体趋势还是吻合的。

- 三角多项式逼近可以看出来效果还是比较好的。说明这么多的原始数据足够拟合出来较好的曲线，证明了线性最小二乘法问题中的确是因为简化了参数。另外，教材中的Matlab程序有错误。在计算参数$a_0$的时候，计算公式应该为

$$a_0 = \frac{2}{N}\sum_{k=1}^{N} f(x_k)cos(0 \times x_k) \tag{29}$$

但是教材中计算$a_0$的k是从$k=0$而不是$k=1$开始，所以多计算了$f(x_k)/N$的函数值，导致函数图象整体向上平移。如果修正了$a_0$的问题，拟合效果就像图(**??**)中那样拟合的较好。

另外从表(**??**)中可以看出来虽然自己算出来的$b_j$的值和Matlab差距较大，但是拟合效果都相差不大。这一点没有找出来原因，因为从表(**??**)看来数据之间差别不大，排除了自编程序的问题。

分析这三幅图片的拟合效果。教材中的程序的方差为7.1460，拟合效果最差；自编程序的方差为0.2236，拟合效果较好；修正$a_0$ 的教材程序方差为0.1553.效果是最好的。说明自编程序还有地方有疏忽。有待进一步改正。

- 贝塞尔曲线的特点就是利用控制曲线产生曲线，由于控制方式的渐变，推测在实际中的图形界面交互中应该是有广泛的应用。

# 7 实验代码

## 7 .1 P202 1

```
  Source.c
  #include <stdio.h>
#include <stdlib.h>
#include "operation.h"
#define _CRT_SECURE_NO_WARNINGS
#pragma warning(disable:4996)
int main()
{
freopen("in2.txt", "r", stdin);
freopen("out2.txt", "w", stdout);
Vector *X, *Y;
Vector *XSquare,*XY;
double Sum_XSquare;
double Sum_XY;
double Sum_X;
double Sum_Y;

X = malloc(sizeof(Vector));
Y = malloc(sizeof(Vector));
XSquare = malloc(sizeof(Vector));
XY = malloc(sizeof(Vector));

/*--calculate the coefficience of regular matrix--*/
InitVector(X,1);
printf("\nX vector\n");
ShowVector(X);

InitVector(Y,1);
printf("\nY vector\n");
ShowVector(Y);

InitVector(XSquare,1);
printf("\nXSquare vector\n");
ShowVector(XSquare);
```

```
InitVector(XY,0);
printf("\nXY vector\n");
ShowVector(XY);

Sum_X = Sum(X);
Sum_Y = Sum(Y);

Square(XSquare);
printf("\nafter square\n");
ShowVector(XSquare);

Sum_XSquare = Sum(XSquare);
VPXY(X, Y, XY);
printf("aaa\n");
ShowVector(XY);
Sum_XY=Sum(XY);
printf("sum_x=%lf\n", Sum_X);
printf("sum_y=%lf\n", Sum_Y);
printf("sum_xsquare=%lf\n", Sum_XSquare);
printf("sum_xy=%lf\n", Sum_XY);

/*--initialize matrixs--*/
double *A, *solution, *B;
A = malloc(4 * sizeof(double));
solution = malloc(2 * sizeof(double));
B = malloc(2 * sizeof(double));
//assign the value to A's element
*(A + 0) = Sum_XSquare;
*(A + 1) = Sum_X;
*(A + 2) = Sum_X;
*(A + 3) = X->N;
*(B + 0) = Sum_XY;
*(B + 1) = Sum_Y;
/*--solve the equation--*/
SolveAXB(A, solution, B);
return 0;
}

operation.h
#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED
/*********************************************************
Module Name: operation.h
Module Date: 11/7/14
Module Auth: Xuanyu Wang
```

```
Description: all operation needed in this question.
************************************************************/
/*----------------Includes---------------*/
#include <math.h>
/*---------Structures and Typedefs-------*/
struct Vector
{
double *X;
int N;
};
typedef struct Vector Vector;
/*---------------Defines-----------------*/


void PrintMatrix(int n_row, int n_column, double *matrix);
void InitVector(Vector *Vec, int jud);
void ShowVector(Vector *Vec);
double Sum(Vector *Vec);
void Square(Vector *Vec);
void VPXY(Vector *X, Vector *Y, Vector *XY);
void SolveAXB(int n_row, int n_column, double *A, double *X, double *B);

/************************************************************
Function Name: InitVector
Function Description: Initialize the variable of vector.
Inputs: Need a point of variable Vector. And a sign jud to judge which kind
           of initializtion will be used.
Outputs: None
************************************************************/
void InitVector(Vector *Vec,int jud)
{
int i;
//Get the number of elements of vector
printf("Input the Number of Node:");
scanf_s("%d", &(Vec->N));
//initialize the vector's elements
Vec->X = malloc((Vec->N)*sizeof(double));
//assign the value of vector's elements'
//There are two modes. if jud=1, get values from stdin
//if jud=0. assign 0 to all elements' value.
for (i = 0; i < Vec->N; i++){
if (jud == 1){
printf("Input the No.%d's value:", i + 1);
scanf_s("%lf", Vec->X + i);
}
else{
```

```
*(Vec->X + i) = 0;
}
}
}
/************************************************************
Function Name: ShowVector
Function Description: Display elements of a vector.
Inputs: Need a point of variable Vector.
Outputs: print elements' value in the stdout
************************************************************/
void ShowVector(Vector *Vec)
{
int i;
for (i = 0; i < Vec->N; i++){
printf("The No.%d's value is:%lf\n", i + 1, *((Vec->X) + i));
}
}
/************************************************************
Function Name:Sum
Function Description: the sum of all elements
Inputs: Need a point of variable Vector.
Outputs: return the result as double
************************************************************/
double Sum(Vector *Vec)
{
int i;
double result = 0;//initlize the sum
for (i = 0; i < Vec->N; i++){
result += *(Vec->X+ i);
}
return result;
}
/************************************************************
Function Name:Square
Function Description: square every element's value
Inputs: Need a point of variable Vector.
Outputs: None
************************************************************/
void Square(Vector *Vec)
{
int i;
for (i = 0; i < Vec->N; i++){
*(Vec->X + i) = pow(*(Vec->X + i), 2);
}
}
/************************************************************
```

```
Function Name:VPXY
Function Description: the element's value of vector XY is corresponded the
X's element times Y's element.
Inputs: Need three points of variable Vector, X,Y,and XY.
Outputs: None
************************************************************/
void VPXY(Vector *X, Vector *Y,Vector *XY)
{
int i;
for (i = 0; i < X->N; i++){
*(XY->X + i) = (*(X->X + i)) * (*(Y->X + i));
}
}

/************************************************************
Function Name: SolveAXB
Function Description: Solve the matrix A*X=B. A is a 2 deminsion matrix.
Inputs:The third is the point of the A. The X & B are 2*1 matrixs.
Outputs: Display the matrix of the X(solution).
Notes:The function need PrintMatrix function. And in this project,
A and B had been eliminated.
************************************************************/
void SolveAXB(double *A, double *X, double *B)
{
int i;
double coe;
//use gauss elemination to change the A and B.
coe = -*(A + 2) / *(A);
*(A + 2) = 0;
*(A + 3) += *(A + 1)*coe;
*(B + 1) += *(B)*coe;
//show A and B
PrintMatrix(2, 2, A);
PrintMatrix(2, 1, B);

//solve the equation.
*(X + 1) = *(B + 1) / *(A + 3);
*(X) = (*(B)-(*(X + 1))* (*(A + 1))) / (*(A + 0));
//show the solution of equation
printf("\nX= ");
PrintMatrix(2, 1, X);
}

/************************************************************
Function Name: PrintMatrix
Function Description: The function display the matrix on the screen.
```

```
The member of each row is equal to the number of column.
Inputs: Three argument are asked. The first is the number of row.
It should be a positive int.
The second is the number of the column and it should be positive int,too.
 The third is the point of the matrix and it should be double.
Outputs: There are no argument need to be return.
Notes:The matrix should be initialized.
*************************************************************/
void PrintMatrix(int n_row, int n_column, double *matrix)
{
printf("\n");
int i, j;//for counting

//display every member of the matrix
for (i = 0; i < n_row; i++){
for (j = 0; j < n_column; j++){
printf("%f   ", *(matrix + i * n_column + j));
            //attention:the format should adjust when you need
}
printf("\n");
}
}
#endif // OPERATION_H_INCLUDED
```

## 7 .2 P215 1

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "operation.h"

int main()
{
    freopen("in.txt","r",stdin);
    freopen("out.txt","w",stdout);
    vector *time,*tmpr;
    vector *matrix;
    vector *Q;
/*--initialize the variables --*/
    time=malloc(sizeof(vector*));
    tmpr=malloc(sizeof(vector*));
    matrix = malloc(sizeof(vector*));
    Q = malloc(sizeof(vector*));
/*--assign every variables every element--*/
    InitVector(time,1);
```

```
    ShowVector(time);
    InitVector(tmpr,1);
    ShowVector(tmpr);
    matrix->num = 9;
    matrix = calcu_coe(matrix, time, tmpr);
    PrintMatrix(3,3,matrix->value);
    Q->num = 3;
    Q = calcu_Q(Q, time, tmpr);
    PrintMatrix(3,1,Q->value);

/*--using LU factorization to solve the equation----*/
    double *P,*idmatrix,*X,*Y,*L,*U;

    P = InitIdMatrix(3, 3, P);
    idmatrix = InitIdMatrix(3, 3, idmatrix);

/*--pivote the matrix--*/
    Pivoting(3, 3, matrix->value,P);
    Q->value = AB(3,3,1,P,Q->value);

/*--Using the triangular factorization to factorize the matrix--*/
//matrix had been changed
    LUFact(3, 3, matrix->value, idmatrix);
    L = idmatrix;
    U = matrix->value;

    X = InitMatrix(3, 1, X,0);
    Y = InitMatrix(3, 1, Y,0);

    BackSub4Y(3, 3, L, Y, Q->value);
    BackSub4X(3, 3, U, X, Y);

    return 0;
}
operation.h

#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED

/************************************************************
Module Name: operation.h
Module Date: 11/15/14
Module Auth: Xuanyu Wang
Description: all operation needed in this question.
************************************************************/
/*----------------Includes---------------*/
```

```
#include <math.h>
/*---------Structures and Typedefs-------*/
struct Vector
{
    int num;
    double *value;
};
typedef struct Vector vector;
/*---------------Defines-----------------*/
//a(m,n) is the true serial number of the element
#define a_mn (i * n_column + j +1)
//the i+(j-i+1) line, i+k column element. i.e.
#define a_jk ((j+1)*n_column+i+k)
#define B 3.267
#define D -0.606


/************************************************************
Function Name: InitVector
Function Description: Initialize the variable of vector.
Inputs: Need a point of variable Vector. And a sign jud to judge
           which kind of initialization will be used.
Outputs: None
************************************************************/
void InitVector(vector *Vec,int jud)
{
int i;
//Get the number of elements of vector
printf("Input the Number of Node:\n");
scanf("%d", &(Vec->num));
//initialize the vector's elements
Vec->value = malloc((Vec->num)*sizeof(double));
    //assign the value of vector's elements'
//There are two modes. if jud=1, get values from stdin
//if jud=0. assign 0 to all elements' value.
for (i = 0; i < Vec->num; i++){
if (jud == 1){
printf("Input the No.%d's value:", i + 1);
scanf("%lf", Vec->value + i);
}
else{
*(Vec->value + i) = 0;
}
}
}
```

```
/************************************************************
 Function Name: ShowVector(Vec *vec)
 Function Description: display the vector's info.
Inputs: need the point of Vec, and the point had been initialized.
 Outputs: No output
************************************************************/
void ShowVector(vector *vec)
{
    int i;
    //for every display the number of it and the value of it.
    for(i = 0; i < vec->num; i++){
        printf("The No.%d value is :%lf \n", i, *((vec->value)+i));
    }
}

/************************************************************
Function Name:Sum
Function Description: the sum of all elements
Inputs: Need a point of variable Vector.
Outputs: return the result as double
************************************************************/
double sum_vector(vector *Vec)
{
    int i,n;
    n=Vec->num;
    double sum=0;
    for(i=0;i<n;i++){
        sum += fabs(*(Vec->value+i));
    }
    return sum;
}
/************************************************************
Function Name:calcu_coe
Function Description: In this case, there are 9 elements in matrix.
            this function calculate every element.
Inputs: Need three points of variable Vector. matrix, x, y.
Outputs: return the point of matrix as vector.
************************************************************/
vector* calcu_coe(vector* matrix, vector* x, vector* y)
{
    //the matrix is a symmetrical matrix, so there are elements pairs.
    matrix->value = malloc(9 * sizeof(double));
    calcu_coe0(matrix->value+0,x,y);
    calcu_coe1(matrix->value+1,x,y);
    calcu_coe2(matrix->value+2,x,y);
    *(matrix->value+3) = *(matrix->value+1);//(2,1)=(1,2)
```

```
    calcu_coe4(matrix->value+4,x,y);
    calcu_coe5(matrix->value+5,x,y);
    *(matrix->value+6) = *(matrix->value+2);//(3,1)=(1,3)
    *(matrix->value+7) = *(matrix->value+5);//(3,2)=(2,3)
    calcu_coe8(matrix->value+8,x,y);

    return (matrix);
}
/************************************************************
Function Name:calcu_Q
Function Description: In this case, there are 3 elements in matrix Q.
            this function calculate every element of Q.
Inputs: Need three points of variable Vector. Q, x, y.
Outputs: return the point of matrix as vector.
************************************************************/
vector* calcu_Q(vector* Q, vector* x, vector* y)
{
    Q->value = malloc(3 * sizeof(double));
    calcu_Q0(Q->value+0,x,y);
    calcu_Q1(Q->value+1,x,y);
    calcu_Q2(Q->value+2,x,y);
    return Q;
}

/************************************************************
Function Name: PrintMatrix
Function Description: The function display the matrix on the screen.
The member of each row is equal to the number of column.
Inputs: Three argument are asked. The first is the number of row.
It should be a positive int.The second is the number of the column
and it should be positive int,too. The third
            is the point of the matrix and it should be double.
Outputs: There are no argument need to be return.
Notes:The matrix should be initialized.
************************************************************/
void PrintMatrix(int n_row,int n_column,double *matrix)
{
    printf("\n");
    int i,j;//for counting

    //display every member of the matrix
    for(i = 0;i < n_row;i++){
        for(j = 0;j < n_column;j++){
        //attention:the format should adjust when you need
            printf("%f   ",*(matrix + i * n_column + j));
        }
```

```
        printf("\n");
    }
}

/*************************************************************
Function Name: Pivoting
Function Description: To find the interchange matrix of matrix with using partial pivoting.
Inputs: The n_row is the number of "matrix"'s rows. It's a positive int.
            The n_column is the number of the column and it's positive int,too.
            the matrix is which would be pivoted. The P is the interchange matrix.
Outputs: None
Notes:
*************************************************************/
void Pivoting(int n_row,int n_column,double *matrix,double *P)
{
    double maxium = 0;
    double *temp;//save the row would be changed
    temp = (double*)malloc(n_column*sizeof(double));
    int i, j;//for counting
    int mark;//save the line which will interchange with current line.
    int fc;//just for interchange two lines of a matrix

    for(i = 0; i < n_column; i++){
//*****************************************************************************
            //for each row,you'll find the most max element of i column
            //in this section, the max will be founded and the line will be marked
            maxium = *(matrix + i*n_column + i);
            mark = i;
            for(j = i+1; j < n_row; j++){//from i line to row-1 line
            //if the max is samller than a(j,i)
                if(fabs(maxium) < fabs(*(matrix+j*n_row+i))){
                    //printf("j = %d\t maxium = %lf\t*(matrix+j*n_row+i) = %lf\n",j,maxium,*
                    maxium = *(matrix+j*n_row+i);//assign the max from a(j,i)
                    mark = j;//the j line will be marked
                }
            }
            //printf("\nmax = %lf\n",maxium);
            //printf("\nthe %d line need to interchange with %d line\n",i+1,mark+1);
            //there are need to interchange with itself.
            if(mark == i){
                continue;
            }
//*****************************************************************
            //then, you need interchange the i line with "mark" line
            for(fc = 0; fc < n_column; fc++){
                *(temp+fc) = *(matrix+mark*n_row+fc);//put a(mk,fc) in cache t(fc)
```

```
                *(matrix+mark*n_row+fc) = *(matrix+i*n_row+fc);//a(mk,fc) = a(i,fc)
                *(matrix+i*n_row+fc) = *(temp +fc);//a(i,fc)=t(fc)
            }
            //To make the P.
            for(fc = 0; fc < n_column; fc++){
                *(temp+fc) = *(P+mark*n_row+fc);//put a(mk,fc) in cache t(fc)
                *(P+mark*n_row+fc) = *(P+i*n_row+fc);//a(mk,fc) = a(i,fc)
                *(P+i*n_row+fc) = *(temp +fc);//a(i,fc)=t(fc)
            }
            //PrintMatrix(n_row,n_column,matrix);
//**************************************************************************
    }
}

/************************************************************
Function Name: InitIdMatrix
Function Description: Produce a matrix which dimension is n_row*n_column
and the diagonal element is 1 and other elements are 0.
Inputs: Three argument are asked. The first is the number of row.
It should be a positive int.
The second is the number of the column and it should
be positive int,too. The third
is the point of the matrix and it should be double.
Outputs: Return the point of the matrix.
Notes:
************************************************************/
double* InitIdMatrix(int n_row,int n_column,double *idmatrix)
{
    int i,j;//counting the row and column
    idmatrix = (double *)malloc((n_row*n_column)*sizeof(double));
    int ii = 1;//when the element in diagonal line, assign the element ii
    int ij = 0;//when the element isn't in diagonal line, assign the element ij
    for(i = 0; i < n_row; i++){//for row
        for(j = 0; j < n_column;j++){//for column
                if(i+1 == j+1){
                    *(idmatrix + a_mn - 1 ) = ii;//give a(i+1,j+i) a value
                }
                else{
                    *(idmatrix + a_mn - 1 ) = ij;
                }
        }
    }
    return (idmatrix);
}

/************************************************************
```

```
Function Name: AB
Function Description: to calculate the result of A times B
Inputs:  The first is the number of A's row. It should be a positive int.
            The second is the number of the A's column and it should be
             positive int,too. The third
            is the number of the B's column.
            The forth is matrix A and the fifth is matrix B
Outputs: Return the result of A*B.
Notes:The function assume A's column is equal with B's row.
*************************************************************/
double* AB(int n_row, int n_column, int single, double *A, double *Q)
{
    printf("\n");
    double *result;//the cache to save the A*B
    result = InitIdMatrix(n_row,n_column,result);//initialize the result
    int i,j,k;//for counting
    double sum = 0;//the sum of a(i,)*b(,j)
    for(i = 0; i < n_row; i++){
        for(j = 0; j < single; j++){
            //calculate the value of ab(i,j)
            for(k = 0; k < n_column; k++){
                sum += (*(A + i*n_row + k)) * (*(Q + k*single + j));
            }
            *(result + i*single + j) = sum;//assign the value to ab(i,j)
            //printf("result = %lf\n\n",*(result + i*single + j));
            sum = 0;//reset the sum for next calculating
        }

    }
    return result;
}

/*************************************************************
Function Name: LUFact
Function Description: The function to find the L and U with triangular factorization
Inputs:The first is the number of row. It should be a positive int.
            The second is the number of the column and it should
            be positive int,too. The third
            is the point of the matrix and it should be double
            and it will be U. The idmatrix is a
            identical matrix.
Outputs: There are no argument need to be return.
Notes:
*************************************************************/
void LUFact(int n_row,int n_column,double *matrix, double *idmatrix)
{
```

```
    int i = 0, j = 0,k = 0;
    double coe;//the coefficient to eliminate
    for(i = 0;i < n_row - 1 ;i++){
        //if the matrix is a 1*1 dimension matrix. There are no need to be factorized.
        if(n_row == 1){
            return;
        }
        //if the matrix need to be factorized.
        for(j = i; j < n_row - 1; j++){
                //find the coefficient for interchange
            coe = -(*(matrix  + i + (j+1)*n_column) / (*(matrix + i * n_column + i)) );
            //printf("coe = %lf\n",coe);
            for(k = 0; k + i < n_column; k++){
                //interchange the matrix and identical matrix to get the U and P
                *(matrix + a_jk) = *(matrix + i * n_column +i + k)
                * coe + (*(matrix + a_jk));
                *(idmatrix + a_jk) = *(idmatrix + i * n_column + i + k)
                * (-coe) + (*(idmatrix  + a_jk));
            }
        }
    }
}

/*************************************************************
Function Name: InitMatrix
Function Description: Produce a matrix which dimension had been input in main().
                                    And the matrix's member is all 0.
Inputs: Three argument are required. The first is the number of row.
 It should be a positive int.
 The second is the number of the column and it should be
  positive int,too. The third
  is the point of the matrix and it should be double.
Outputs: Return a point of the matrix.
Notes:The module is individual
*************************************************************/
double* InitMatrix(int n_row,int n_column,double *matrix,int special)
{
    if(special == 0){
        int i,j;//counting the row and column
        matrix = (double *)malloc((n_row*n_column)*sizeof(double));
        double value = 0;//define the matrix's value
        for(i = 0;i < n_row;i++){//for row
            for(j = 0;j < n_column;j++){//for column
                *(matrix + i* n_column + j ) = value;//give a(i+1,j+i) a value
            }
        }
```

```
        return (matrix);//return a one-dimension double array
    }
    else{
        int i,j;//counting the row and column
        matrix = (double *)malloc((n_row*n_column)*sizeof(double));
        //int count = 0;//initialize the matrix's value
        for(i = 0; i < n_row; i++){//for row
            for(j = 0; j < n_column;j++){//for column
                *(matrix + a_mn - 1 ) = pow((double)(i+1),(double)(j));
            }
        }
        return (matrix);
    }
}

/************************************************************
Function Name: BackSub4Y
Function Description: solve the Y in LY=B.
Inputs:The first is the number of row. It should be a positive int.
            The second is the number of the column and it should be positive int,too.
Outputs: There are no argument need to be return.
Notes:
************************************************************/
void  BackSub4Y(int n_row, int n_column, double *L, double *Y, double *Q)
{
    int i = 0, j = 0;
    double sum_0i = 0;//the sum of l(i,j)*y(j), j is from 0 to i.
    for(i = 0; i < n_column; i++){
            //get the sum
        for(j = 0; j < i; j++){
            sum_0i += *(L + a_mn - 1) * (*(Y+ j));
        }
    //solve the Y(i)
        *(Y + i) = (*(Q + i) - sum_0i) / *(L + i*n_column + i);
        sum_0i = 0;//reset the sum for next calculation
    }
    printf("Y = ");
    PrintMatrix(n_row, 1, Y);
}

/************************************************************
Function Name: BackSub4X
Function Description: solve the X in UX=Y.
Inputs:The first is the number of row. It should be a positive int.
            The second is the number of the column and it should be positive int,too.
Outputs: There are no argument need to be return.
```

```
Notes:
************************************************************/
void BackSub4X(int n_row, int n_column, double *U, double *X, double *Y)
{
    int i = 0, j = 0;
    double sum_iend = 0;//the sum of u(i,j)*x(j), j is from end to i.
    for(i = n_row - 1; i >= 0; i--){
            //calculate the sum
        for(j = n_column - 1; j > i; j--){
            sum_iend += *(U + a_mn - 1) * (*(X+ j));
        }
    //calculate the X(i)
        *(X + i) = (*(Y + i) - sum_iend) / *(U + i*n_column + i);
        sum_iend = 0;//reset the sum for next calculation
    }
    printf("X= ");
    PrintMatrix(n_row, 1, X);
}

/*********************************************************
Function Name:calcu_coe0
Function Description: calculate the (1,1)of the matrix
Inputs: Need three variables. coe, x, y. coe is a point of double.
        x,y are points vector.
Outputs: None
*************************************************************/
void calcu_coe0(double* coe, vector* x,vector *y)
{
    int i;
    *coe = 0;
    for(i = 0; i < x->num; i++){
        *coe += pow(cos(B* (*(x->value+i)) ), 2);
    }
}
/*********************************************************
Function Name:calcu_coe1
Function Description: calculate the (1,2)of the matrix
Inputs: Need three variables. coe, x, y. coe is a point of double.
        x,y are points vector.
Outputs: None
*************************************************************/
void calcu_coe1(double* coe, vector* x,vector *y)
{
    int i;
    *coe = 0;
    for(i = 0; i < x->num; i++){
```

```
        *coe += cos(B * (*(x->value+i)) ) * sin(D * (*(x->value+i)) );
    }
}
/************************************************************
Function Name:calcu_coe2
Function Description: calculate the (1,3)of the matrix
Inputs: Need three variables. coe, x, y. coe is a point of double.
        x,y are points vector.
Outputs: None
************************************************************/
void calcu_coe2(double* coe, vector* x,vector *y)
{
    int i;
    *coe = 0;
    for(i = 0; i < x->num; i++){
        *coe += cos(B * (*(x->value+i)) );
    }
}
/************************************************************
Function Name:calcu_coe4
Function Description: calculate the (2,1)of the matrix
Inputs: Need three variables. coe, x, y. coe is a point of double.
        x,y are points vector.
Outputs: None
************************************************************/
void calcu_coe4(double* coe, vector* x,vector *y)
{
    int i;
    *coe = 0;
    for(i = 0; i < x->num; i++){
        *coe += pow(sin(D * (*(x->value+i)) ), 2);
    }
}
/************************************************************
Function Name:calcu_coe5
Function Description: calculate the (2,3)of the matrix
Inputs: Need three variables. coe, x, y. coe is a point of double.
        x,y are points vector.
Outputs: None
************************************************************/
void calcu_coe5(double* coe, vector* x,vector *y)
{
    int i;
    *coe = 0;
    for(i = 0; i < x->num; i++){
        *coe += sin(D * (*(x->value+i)) );
```

```
    }
}
/************************************************************
Function Name:calcu_coe8
Function Description: calculate the (3,3)of the matrix
Inputs: Need three variables. coe, x, y. coe is a point of double.
        x,y are points vector.
Outputs: None
************************************************************/
void calcu_coe8(double* coe, vector* x,vector *y)
{
    int i;
    *coe = 0;
    for(i = 0; i < x->num; i++){
        *coe += 1;
    }
}
/************************************************************
Function Name:calcu_coeQ0
Function Description: calculate the (1,1)of the Q
Inputs: Need three variables. coe, x, y. coe is a point of double.
        x,y are points vector.
Outputs: None
************************************************************/
void calcu_Q0(double* coe, vector* x, vector* y)
{
    int i;
    *coe = 0;
    for(i = 0; i< x->num; i++){
        *coe += cos(B * *(x->value+i)) * *(y->value+i);
    }
}

/************************************************************
Function Name:calcu_coeQ1
Function Description: calculate the (2,1)of the Q
Inputs: Need three variables. coe, x, y. coe is a point of double.
        x,y are points vector.
Outputs: None
************************************************************/
void calcu_Q1(double* coe, vector* x, vector* y)
{
    int i;
    *coe = 0;
    for(i = 0; i< x->num; i++){
        *coe += sin(D * *(x->value+i)) * *(y->value+i);
```

```
    }
}

/************************************************************
Function Name:calcu_coeQ2
Function Description: calculate the (3,1)of the Q
Inputs: Need three variables. coe, x, y. coe is a point of double.
        x,y are points vector.
Outputs: None
************************************************************/
void calcu_Q2(double* coe, vector* x, vector* y)
{
    int i;
    *coe = 0;
    for(i = 0; i< x->num; i++){
        *coe += *(y->value+i);
    }
}
#endif // OPERATION_H_INCLUDED
```

## 7 .3 P229 1

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "operation.h"


int main()
{
    freopen("in.txt","r",stdin);
    //freopen("test.txt","r",stdin);
    freopen("outin.txt","w",stdout);
    vector *d, *h, *m, *u;
    vector *Xnode, *Ynode;
    double **S;
    double ds_a = 0;
    double ds_b = 98;
    int i;
/*--initialize the vectors--*/
    d = init_vector(d,0);
    h = init_vector(h,0);
    m = init_vector(m,0);
    u = init_vector(m,0);
    Xnode = init_vector(Xnode,1);
```

```
    Ynode = init_vector(Ynode,1);
    S = malloc((Xnode->N-1) * sizeof(double*));
/*--calculate every element of every vector--*/
    calcu_h(h,Xnode);
    calcu_d(d,Xnode,Ynode);
    calcu_u(d,u,Xnode);
    printf("\nh\n");
    ShowVector(h);
    printf("\nd\n");
    ShowVector(d);
    printf("\nm\n");
    ShowVector(m);
    printf("\nu\n");
    printf("\nu\n");
    ShowVector(u);
    //ShowVector(X);
    //ShowVector(Y);
/*--using Gauss-Seidel iteration to solve the equation HM=V--*/
    calcu_m(m,d,u,h,Xnode,ds_a,ds_b);
    *(m->value+0) = (3 / *(h->value)) * (*(d->value) - ds_a) - *(m->value+1) / 2;
    *(m->value+Xnode->N - 1) = (3 / *(h->value+Xnode->N - 2)) *
                (ds_b - *(d->value+Xnode->N - 2)) - *(m->value+Xnode->N - 2) / 2;
    printf("\nm:\n");
    ShowVector(m);
/*--use what have calculated to calculate the coefficient of clamped cubic spline--*/
    calcu_S(S,Ynode,d,h,m);
/*--show the value of coefficients--*/
    for(i = 0; i < Ynode->N-1; i++){
        printf("%lf&\t%lf&\t%lf&\t%lf\n",*(*(S+i)+0),*(*(S+i)+1),*(*(S+i)+2),*(*(S+i)+3));
    }
    return 0;
}
```

operation.h

```
#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED
/*************************************************************
Module Name: operation.h
Module Date: 101/13/2014
Module Auth: Xuanyu Wang

Description: To produce,display a matrix.
    Revision History: None
*************************************************************/
/*----------------Includes---------------*/
```

```
//the head files that were included
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

/*---------Structures and Typedefs-------*/
struct v{
    int N;
    double *value;
};
typedef struct v vector;

/*---------------Defines-----------------*/
#define a_mn (i * n_column + j +1)//a(m,n) is the true serial number of the element
#define a_jk ((j+1)*n_column+i+k)//the i+(j-i+1) line, i+k column element. i.e.


double* InitMatrix(int n_row,int n_column,double *matrix,int special);
double* TridiagMatrix(int n_row,int n_column,double *matrix,vector *h);
double* InitX(int n_row,int n_column,double *matrix);
void PrintMatrix(int n_row,int n_column,double *matrix);
double* AB(int n_row, int n_column, int single, double *A, double *B);
double* InitIdMatrix(int n_row,int n_column,double *idmatrix);

/************************************************************
Function Name: InitVector
Function Description: Initialize the variable of vector.
Inputs: Need a point of variable Vector. And a sign jud to judge which kind
           of initializtion will be used.
Outputs: None
************************************************************/
vector* init_vector(vector *v,int jud)
{
    int i;
    v = malloc(sizeof(vector));

    if(jud==0){
        printf("Input the Number of Node:\n");
        scanf("%d",&(v->N));
        v->value = malloc((v->N)*sizeof(double));
        return v;
    }
    if(jud==1){
        printf("Input the Number of Node:\n");
        scanf("%d",&(v->N));
        v->value = malloc((v->N)*sizeof(double));
```

```
        for(i = 0; i < v->N; i++){
            printf("Input No.%d's value: ");
            scanf("%lf",v->value+i);
        }
        return v;
    }
}

/************************************************************
 Function Name: ShowVector(Vec *vec)
 Function Description: display the vector's info.
Inputs: need the point of Vec, and the point had been initialized.
 Outputs: No output
************************************************************/
void ShowVector(vector *vec)
{
    int i;
    //for every display the number of it and the value of it.
    for(i = 0; i < vec->N; i++){
        printf("The No.%d value is :%lf \n", i, *((vec->value)+i));
    }
    printf("\n");
}

/************************************************************
 Function Name: calcu_h
 Function Description: calculate the h_k for clamped cubic spline.
Inputs: h and X are points of vector. X is the node.
 Outputs: No output
************************************************************/
void calcu_h(vector *h,vector *X)
{
    int i;
    for(i = 0;i < h->N; i++){
        *(h->value + i) = *(X->value + i + 1) - *(X->value + i);
    }
}
/************************************************************
 Function Name: calcu_d
 Function Description: calculate the d_k for clamped cubic spline.
Inputs: h ,Y and X are points of vector. X and Y are the nodes.
 Outputs: No output
************************************************************/
void calcu_d(vector *d, vector *X, vector *Y)
{
    int i;
```

```
    for(i = 0; i < d->N; i++){
        *(d->value + i) = (*(Y->value + i+1) - *(Y->value + i))
                            / (*(X->value + i+1) - *(X->value + i));
    }
}
/*************************************************************
 Function Name: calcu_u
 Function Description: calculate the u_k for clamped cubic spline.
Inputs: h ,u and X are points of vector. X is nodes.
 Outputs: No output
*************************************************************/
void calcu_u(vector *d, vector *u, vector *X)
{
    int i;
    for(i = 1; i <= X->N - 2; i++){
        *(u->value+i) = 6 * (*(d->value+i) - *(d->value+i-1));
    }
}
/*************************************************************
 Function Name: calcu_S
 Function Description: calculate the S(i,j) for clamped cubic spline.
Inputs: h ,Y and d,m are points of vector.S is the point of double's point.
 Outputs: No output
*************************************************************/
void calcu_S(double **S, vector *Y, vector *d, vector *h, vector *m)
{
    int i, j;
    for(i = 0; i < Y->N-1; i++){
        *(S+i) = malloc( 4 * sizeof(double));
        *(*(S+i)+0) = *(Y->value+i);
        *(*(S+i)+1) = *(d->value+i) - *(h->value+i) *
                (2* (*m->value+i) + *(m->value+i+1))/6;
        *(*(S+i)+2) = *(m->value+i)/2;
        *(*(S+i)+3) = (*(m->value+i+1)-*(m->value+i))/(6*(*(h->value+i)));
    }
}

/*************************************************************
 Function Name: calcu_m
 Function Description: calculate the m_k for clamped cubic spline.
Inputs: m, d, u, h, X are points of vector. ds_a and ds_b is the difference of endpoint.
 Outputs: No output
*************************************************************/
void calcu_m(vector *m, vector *d, vector *u, vector *h,
                                vector *Xnode,double ds_a,double ds_b)
{
```

```
    double *matrix, *X, *B,*result;
    int n_row,n_column;
    int single = 1;
    int i;

    n_row = (Xnode->N) - 2;
    n_column = (Xnode->N) - 2;


    matrix = InitMatrix(n_row,n_column,matrix,0);//produce a matrix
    X = InitX(n_row, single, X);
    B = InitMatrix(n_row, single, B,0);
    result = InitMatrix(n_row,single,result,0);

    for(i = 0; i < n_row; i++){
        if(i == 0){
            printf("i=%d,in 1\n",i);
            *(B+i) = *(u->value+1) - 3*( *(d->value+0) -  ds_a);
        printf("B+%d=%lf\n",i,*(B+i));
            //printf("\n(%lf+1) - 3*( *(%lf+0) -  0.2) = %lf\n",
            *(u->value),*(d->value+0),*(B+i));
        }
        else if(1<=i || i<=n_row - 1){
            printf("i=%d,in 2\n",i);
            *(B+i) = *(u->value + i+1);
            printf("B+%d=%lf\n",i,*(B+i));
        }
        if(i == n_row - 1){
            printf("i=%d,in 3\n",i);
            *(B+i) = *(u->value+n_row) - 3*(ds_b - *(d->value+n_row));
            printf("B+%d=%lf\n",i,*(B+i));
        }

    }
    //printf("Matrix = ");
    //PrintMatrix(n_row,n_column,matrix);//display the matrix
    //printf("\nX = ");
    //PrintMatrix(n_row, single, X);
    //printf("\nB = ");
    //PrintMatrix(n_row, single, B);

//change the matrix into a tridiagonal matrix
    matrix = TridiagMatrix(n_row,n_column,matrix,h);
    printf("\nTriMatrix = ");
    PrintMatrix(n_row,n_column,matrix);//display the matrix
    printf("\nB = ");
```

```
    PrintMatrix(n_row, 1, B);//display the matrix

    GSIter4tri(n_row, n_column,matrix, B, X);
    for(i = 1; i <= n_row; i++){
            *(m->value+i) = *(X+i-1);
    }
    //ShowVector(m);
    result = AB(n_row,n_column,single,matrix,X);
    PrintMatrix(n_row,single,result);
}

/*************************************************************
Function Name: AB
Function Description: to calculate the result of A times B
Inputs:  The first is the number of A's row. It should be a positive int.
            The second is the number of the A's column and it should be
            positive int,too. The third
            is the number of the B's column. The forth is matrix A
            and the fifth is matrix B
Outputs: Return the result of A*B.
Notes:The function assume A's column is equal with B's row.
*************************************************************/
double* AB(int n_row, int n_column, int single, double *A, double *B)
{
    printf("\n");
    double *result;//the cache to save the A*B
    result = InitIdMatrix(n_row,n_column,result);//initialize the result
    int i,j,k;//for counting
    double sum = 0;//the sum of a(i,)*b(,j)
    for(i = 0; i < n_row; i++){
        for(j = 0; j < single; j++){
            //calculate the value of ab(i,j)
            for(k = 0; k < n_column; k++){
                sum += (*(A + i*n_row + k)) * (*(B + k*single + j));
            }
            *(result + i*single + j) = sum;//assign the value to ab(i,j)
            //printf("result = %lf\n\n",*(result + i*single + j));
            sum = 0;//reset the sum for next calculating
        }

    }
    return result;
}

/*************************************************************
Function Name: InitIdMatrix
```

```
Function Description: Produce a matrix which dimension is n_row*n_column
                                        and the diagonal element is 1 and other elements are
Inputs: Three argument are asked. The first is the number of row. It should be a positive in
            The second is the number of the column and it should be positive int,too. The th
            is the point of the matrix and it should be double.
Outputs: Return the point of the matrix.
Notes:
*************************************************************/
double* InitIdMatrix(int n_row,int n_column,double *idmatrix)
{
    int i,j;//counting the row and column
    idmatrix = (double *)malloc((n_row*n_column)*sizeof(double));
    int ii = 1;//when the element in diagonal line, assign the element ii
    int ij = 0;//when the element isn't in diagonal line, assign the element ij
    for(i = 0; i < n_row; i++){//for row
        for(j = 0; j < n_column;j++){//for column
                if(i+1 == j+1){
                    *(idmatrix + a_mn - 1 ) = ii;//give a(i+1,j+i) a value
                }
                else{
                    *(idmatrix + a_mn - 1 ) = ij;
                }
        }
    }
    return (idmatrix);
}

/*************************************************************
Function Name: InitX
Function Description: Produce a matrix which dimension
 had been input in main().
And the matrix's member is all 0.
Inputs: Three argument are asked. The first is the number of row.
It should be a positive int.
The second is the number of the column and it
should be positive int,too. The third
is the point of the matrix and it should be double.
Outputs: Return a point of the matrix.
Notes:
*************************************************************/
double* InitX(int n_row,int n_column,double *matrix)
{
    int i,j;//counting the row and column
    matrix = (double *)malloc((n_row*n_column)*sizeof(double));
    double coun = 0;//initialize the matrix's value
    for(i = 0;i < n_row;i++){//for row
```

```
        for(j = 0;j < n_column;j++){//for column
            *(matrix + i* n_column + j ) = coun;//give a(i+1,j+i) a value
        }
    }
    return (matrix);
}

/************************************************************
Function Name: PrintMatrix
Function Description: The function display the matrix on the screen. The member of
each row is equal to the number of column.
Inputs: Three argument are asked. The first is the number of row. It should be a positive in
The second is the number of the column and it should be positive int,too. The third
is the point of the matrix and it should be double.
Outputs: There are no argument need to be return.
Notes:The matrix should be initialized.
************************************************************/
void PrintMatrix(int n_row,int n_column,double *matrix)
{
    printf("\n");
    int i,j;//for counting

    //display every member of the matrix
    for(i = 0;i < n_row;i++){
        for(j = 0;j < n_column;j++){
            printf("%.3lf   ",*(matrix + i * n_column + j));
        }
        printf("\n");
    }
}

/************************************************************
Function Name: GSIter
Function Description: Use the Gauss-Seidel iteration to solve the matrix euqation.
Inputs: Three argument are asked. The first is the number of row.
It should be a positive int.
The second is the number of the column and it should be positive int,too. The third
is the point of the matrix and it should be double. Generally, the matrix is a
n*n matrix and the X and B are n*1 matrix.
Outputs: Print the solution X.
Notes:
************************************************************/
void GSIter4tri(int n_row, int n_column, double *A, double *B, double *X)
{
    int i,j,k,single = 1;
    double sum,tol = 0.000000001,err;
```

```
    double *te;//cache for the older solution
    //PrintMatrix(n_row,n_column,A);
    te = InitX(n_row,single,te);
    //the circle will be ended while the error<tolerance
    do{
        for(k = 0;k < n_column;k++){
            *(te+k) = *(X+k);
        }
        //calculate the sum
        for(i = 0; i < n_row; i++){
            for(j = i-1; j < i+2; j++){
                    if((j>=0)&&(j<n_column)&&(j != i) ){
                        sum += *(X + j) * (*(A + i*n_column + j));
                    }
            }
        //calculate the X(i)
            *(X + i) = ((*(B + i)) - sum) / (*(A + i*n_column + i));
            sum = 0;
        }
        err = 0;
        //calculate the error
        for(k = 0;k < n_column;k++){
            err += fabs(*(X+k)-*(te+k));
        }

    }while(err > tol);
    printf("X = ");
    PrintMatrix(n_row, 1, X);
}



/*************************************************************
Function Name: InitMatrix
Function Description: Produce a matrix which dimension had been input in main().
And the matrix's member is all 0.
Inputs: Three argument are required. The first is the number of row.
It should be a positive int.The second is the number of the column
and it should be positive int,too. The third
is the point of the matrix and it should be double.
Outputs: Return a point of the matrix.
Notes:The module is individual
*************************************************************/
double* InitMatrix(int n_row,int n_column,double *matrix,int special)
{
    if(special == 0){
```

```
        int i,j;//counting the row and column
        //allocate the space for the matrix
        matrix = (double *)malloc((n_row*n_column)*sizeof(double));
        double value = 0;//define the matrix's value
        for(i = 0;i < n_row;i++){//for row
            for(j = 0;j < n_column;j++){//for column
                *(matrix + i* n_column + j ) = value;//give a(i+1,j+i) a value
            }
        }
        return (matrix);//return a one-dimension double array
    }
    else{
        int i,j;//counting the row and column
        //allocate the space for the matrix
        matrix = (double *)malloc((n_row*n_column)*sizeof(double));
        //int count = 0;//initialize the matrix's value
        for(i = 0; i < n_row; i++){//for row
            for(j = 0; j < n_column;j++){//for column
            //give a(i+1,j+i) a value
                *(matrix + a_mn - 1 ) = pow((double)(i+1),(double)(j));
            }
        }
        return (matrix);
    }
}

/************************************************************
Function Name: TridiagMatrix
Function Description: Produce a matrix which is a tridiagonal matrix.
Inputs: Three argument are asked. The first is the number of row.
 It should be a positive int. The second is the number of the column
 and it should be positive int,too. The third
is the point of the matrix and it should be double.
Outputs: Return a point of the matrix.
Notes: The matrix should be initialized.
************************************************************/
double* TridiagMatrix(int n_row,int n_column,double *matrix,vector *h)
{
    int i;//i is for counting
    for(i = 0;i < n_row;i++){
        //for the first row, we need two coefficients
        if((i + 1) == 1){
            printf("first row:input the coefficient of
             the a(%d,%d) & a(%d,%d):\n",1,1,1,2);
            //scanf("%lf %lf",matrix,(matrix+1));
            *(matrix) = 3*(*(h->value))/2 + 2*(*(h->value+1));
```

```
            *(matrix + 1) = *(h->value+1);
        }
        //for the last row, we need two coefficients
        else if((i + 1) == n_row){
            printf("input the coefficient of the a(%d,%d) &
                           a(%d,%d):\n",n_row,n_row-1,n_row,n_row);
            //scanf("%lf %lf",matrix+n_row*n_column-2,matrix+n_row*n_column-1);
            *(matrix+n_row*n_column-2) = *(h->value+n_row-1);
            *(matrix+n_row*n_column-1) = 2 * (*(h->value+n_row-1))
                                + 3 * (*(h->value+n_row))/2;
        }
        //for each row except first and last need three coefficient
        else{
            printf("input the coefficient of the a(%d,%d) & a(%d,%d)
                            & a(%d,%d):\n",i+1,i,i+1,i+1,i+1,i+2);
            *(matrix+i*n_column+i-1) = *(h->value+i);
            *(matrix+i*n_column+i) = 2*(*(h->value+i) + *(h->value+i+1));
            *(matrix+i*n_column+i+1) = *(h->value+i+1);
        }
    }
    return (matrix);//return the point of one-dimension array
}


#endif // OPERATION_H_INCLUDED
```

## 7 .4 P245 3

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "operation.h"
#define PI 3.1415926

int main()
{
    freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    vector *node_x, *node_y, *X4coe;
    vector *coe_a, *coe_b;
/*--initialize the variables--*/
    node_x = init_vector(node_x,1);
    node_y = init_vector(node_y,1);
    coe_a = init_vector(coe_a,0);
    coe_b = init_vector(coe_b,0);
    X4coe = change_nodex2X(node_x,X4coe);
```

```
/*--calculate the coefficients of polynomial--*/
    calcu_coe_a(coe_a,X4coe,node_y);
    printf("coe_a:\n");
    ShowVector(coe_a);

    calcu_coe_b(coe_b,X4coe,node_y);
    printf("coe_b:\n");
    ShowVector(coe_b);
/*--display the polynomial--*/
    show_polynomial(coe_a,coe_b);
    return 0;
}
```

operation.h

```
#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED
/************************************************************
Module Name: operation.h
Module Date: 101/13/2014
Module Auth: Xuanyu Wang

Description: To produce,display a matrix.
    Revision History: None
************************************************************/
/*----------------Includes---------------*/
//the head files that were included
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define PI 3.141592653589793

/*---------Structures and Typedefs-------*/
struct v{
    int N;
    double *value;
};
typedef struct v vector;

/*---------------Defines-----------------*/

/************************************************************
Function Name: InitVector
Function Description: Initialize the variable of vector.
Inputs: Need a point of variable Vector. And a sign jud to judge which kind
```

```
           of initializtion will be used.
Outputs: None
************************************************************/
vector* init_vector(vector *v,int jud)
{
    int i;
    v = malloc(sizeof(vector*));

    if(jud==0){
        //printf("Input the Number of Node:\n");
        scanf("%d",&(v->N));
        v->value = malloc((v->N)*sizeof(double));
        return v;
    }
    if(jud==1){
        //printf("Input the Number of Node:\n");
        scanf("%d",&(v->N));
        v->value = malloc((v->N)*sizeof(double));
        for(i = 0; i < v->N; i++){
            printf("Input No.%d's value: ",i);
            scanf("%lf",v->value+i);
        }
        return v;
    }
}

/************************************************************
 Function Name: ShowVector(Vec *vec)
 Function Description: display the vector's info.
Inputs: need the point of Vec, and the point had been initialized.
 Outputs: No output
************************************************************/
void ShowVector(vector *vec)
{
    int i;
    //for every display the number of it and the value of it.
    for(i = 0; i < vec->N; i++){
        //printf("The No.%d value is :%lf \n", i, *((vec->value)+i));
                printf("$a_%d$ & %lf \\\\\n", i+1, *((vec->value)+i));
    }
    printf("\n");
}

/************************************************************
 Function Name: change_nodex2X
 Function Description: change the x to x' to use trigonometric polynomial.
```

```
Inputs: node_x is the original value, and X is value had been mapped.
 Outputs: No output
************************************************************/
vector* change_nodex2X(vector *node_x, vector *X)
{
    int i;
    X = malloc(sizeof(vector*));
    X->N = node_x->N;
    X->value = malloc((X->N-1) * sizeof(double));
    for(i = 0;i < node_x->N; i++){
        *(X->value+i) = -PI + i*(2*PI)/(node_x->N-1);
    }
    return X;
}

/************************************************************
 Function Name: calcu_coe_a
 Function Description: calculate the coefficients a_j of trigonometric polynomial
Inputs: X4coe is the value for trigonometric polynomial.
        coe_a is the coefficient of polynomial. node_y are value of f(x_k)
 Outputs: No output
************************************************************/
void calcu_coe_a(vector *coe_a,vector *X4coe, vector *node_y)
{
    int j, k,N;
    double sum = 0;
    N = X4coe->N-1;

    for(j = 0; j < coe_a->N; j++){
        sum = 0;
        for(k = 1; k <= N; k++){
            sum += *(node_y->value+k) * cos( j * (*(X4coe->value+k)));
        }
        if(j==0){
            sum+=*(node_y->value+0);
        }
        *(coe_a->value+j) = (2* sum/N) ;
    }
    *(coe_a->value+0) /= 2;
}
/************************************************************
 Function Name: calcu_coe_b
 Function Description: calculate the coefficients b_j of trigonometric polynomial
Inputs: X4coe is the value for trigonometric polynomial.
        coe_b is the coefficient of polynomial. node_y are value of f(x_k)
 Outputs: No output
```

```
************************************************************/
void calcu_coe_b(vector *coe_b,vector *X4coe, vector *node_y)
{
    int j, k,N;
    double sum = 0;
    N = X4coe->N;
    for(j = 1; j < coe_b->N; j++){
        sum = 0;
        for(k = 1; k < N; k++){
            sum += *(node_y->value+k) * sin( j * (*(X4coe->value+k)));
        }
        *(coe_b->value+j) = (2* sum/N) ;
    }
}
/************************************************************
 Function Name: show_polynomial
 Function Description: display the polynomial
Inputs: coe_a and coe_b
 Outputs: No output
************************************************************/
void show_polynomial(vector *coe_a,vector *coe_b)
{
    int j;
    printf("\n%lf",*(coe_a->value) / 2);
    for(j = 1; j <= coe_b->N; j++){
        printf("+");
        printf("(%lf*cos(%d*x*pi/12) + (%lf)*sin(%d*x*pi/12))",
        *(coe_a->value+j),j,*(coe_b->value+j),j);
    }
}
#endif // OPERATION_H_INCLUDED
```

## 7 .5 P245 3

```
syms t;
h=[0:.001:1];
% x=3*t+10.5*t.^2-5.5*t.^3
x1=3*t*(1-t).^2+3*(t.^2)*(1-t)+3*(t.^3)
y1=2*3*t*(1-t)^2+3*t^2*(1-t)
x2=3*(1-4*t+6*t^2-4*t^3+t^4)+4*(4*t-12*t^2+12*t^3-4*t^4)
+5*(6*t^2-12*t^3+6*t^4)+6*(4*t^3-4*t^4)+7*t^4
y2=-1*(4*t-12*t^2+12*t^3-4*t^4)-2*(6*t^2-12*t^3+6*t^4)+(4*t^3-4*t^4)
x3=7*(1-t)^2+4*3*t*(1-t)^2+2*3*t^2*(1-t)
y3=-3*3*t*(1-t)^2-3*t^2*(1-t)
h1=plot(subs(x1,h),subs(y1,h),'r')
```

```
hold on
h2=plot(subs(x2,h),subs(y2,h),'g')
hold on
h3=plot(subs(x3,h),subs(y3,h),'b')
legend([h1,h2,h3],'第一个控制点集构造的贝塞尔曲线','第二个控制点集
构造的贝塞尔曲线','第三个控制点集构造的贝塞尔曲线','Location','northeast')
xlabel('x')
ylabel('y')
```