浙江大学

Numerical Analysis –Summer 2016

《数值计算方法》课程大作业

姓　　名：李浩然

学　　号：3140102316

手　　机：178-1689-0200

指导教师：刘山

2016 年 6 月

目录

基于内插和近似方法的儿童发育速查应用

（李浩然 3140102316）

1 绪论

根据儿童体格发育调查结果，国家卫生部妇社司组织相关专家，研究制定了《中国 7 岁以下儿童生长发育参照标准》（以下简称《参照标准》）该标准已于 2009 年 6 月 2 日由卫生部正式公布(男孩和女孩的发育情况不同，具体可参照附件表格中的数据)。

为了方便儿童家长了解自己的孩子的生长发育状况是否正常，本文将以此为目标进行探索。

2 设计目标

2.1 总体目标

本文将针对任意年龄（0~81 个月内）的小孩，通过分析建模，基于附件中的《中国 7 岁以下儿童生长发育参照标准》中的男女儿童的身高体重的数据，通过插值与拟合，利用 *Matlab* 实现并设计一个应用，实现快速查询标准身高和体重。即输入小孩的生日（比如 2013/12/15），性别，输出其标准身高和体重，并保证应用的客户友好性，操作简便，界面美观。在代码中提供详细注释，以便于程序优化。

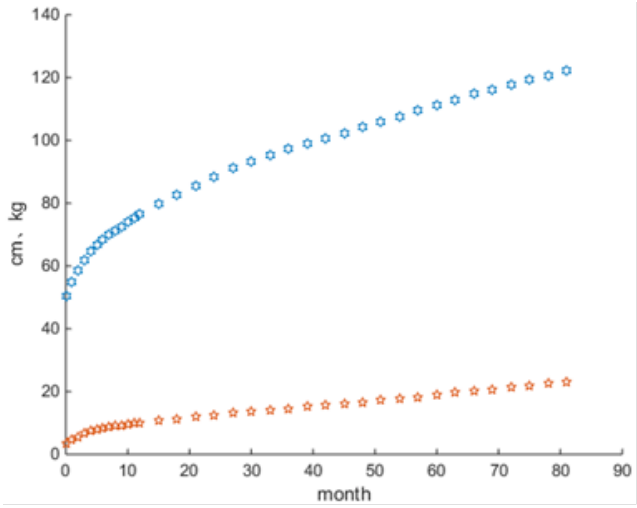
2.2 总体假设

因为 1、2 天对于儿童身高体重的环境因素的影响远大于儿童正常生长发育的影响，因此在本题中以月为单位，将拟合\插值出空缺月份的数据。

3 设计思路

3.1 预测问题的解决

附件中给出了 0~81 月龄的男女儿童的标准身高体重值，需要求出期间任意年龄的儿童的标准身高和体重，因此选取 *month* 为自变量*(x)*，*cm*，*kg* 为自变量 *y*，绘制中位数的 *cm-month* 与 *kg-month* 散点图。



从图中可以看到，数据点基本呈现一次上升趋势，表明统计的误差和不确定因素的影响较小。另外数据点分布比较均匀。

所以大体方法是（1）使用一条曲线来近似点之间的关系；（2）分析曲线的近似程度和精度，（3）找出任意 *month* 的 *cm* 与 *kg* 值；（4）尽量给出一个预测的置信区间

3.2 插值和拟合的选择

插值方法在数据点处于曲线完全重合，适用于数据精确的场合；而本课题研究的问题，因为是实验采样数据，而标准值就是该年龄儿童应该处于的身高或者体重的标准值，是一个精确的数字，应该通过数据点，因此采用拟合不够严密，不适合使用拟合，应该使用插值方法来求出大致关系。

但是为了增强对插值和拟合的认识和应用，本文亦先使用了插值方法进行编程和分析，再使用拟合方法进行分析。

4 算法描述

4.1 插值方法的编写和分析

4.1.1 概述

插值可用的方法主要有：Lagrange 插值，Newton 插值，分段低次插值，样条插值等等。由于 Lagrange 插值和 Newton 插值的最终结果是一样的，所以仅以较优的 Newton 插值为代表进行编程。

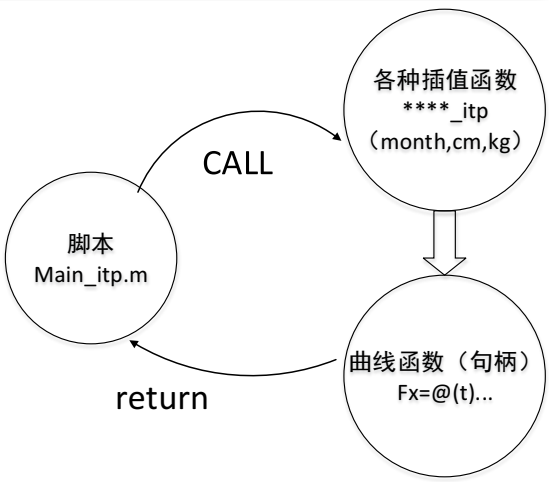
首先编写脚本 Main_itp.m，用来输入数据，调用插值方法，绘制曲线和数据点图。

(在此仅以男孩的标准身高为例子，其他的算法相同，在此不再赘述)。

- **Main_itp.m**

```
%Main_itp.m interpolating, plotting and calc.f(10)
% Given data
load('month.mat');
load('cm_b.mat');
% Interpolating by diff. methods
%   remove one of comment mark (%) below to use that method

% Interpolating by diff. methods
%    remove one of comment mark (%) below to use that method
% {
%     t=month;month=cm_b;cm_b=t;           % use Inverse
interpolation
%     f=Newton_itp(month,cm_b);      % use newton method (polynomial)
    f=Piecewise_itp(month,cm_b,1); % \
%     f=Piecewise_itp(month,cm_b,5); % |use piecewise low-order
newton
%     f=Piecewise_itp(month,cm_b,7); % |
%     f=Piecewise_itp(month,cm_b,35); % /  (same as Newton_itp)
%     f=CubicN_itp(month,cm_b);      % use Cubic Spline interpolation
%     f=Polynomial_fit(month,cm_b,2);% use 2-degree fitting
% }

% Plotting
x = 0:0.1:80; y=f(x);
figure;hold on;
plot(x,y,'r','linewid',1.5);
plot(month,cm_b,'b*');
legend(gca,'Interpolation','cm_b vs.
month','Location','NorthWest')
xlabel 'month';ylabel 'cm_b';
title 'Interpolating';
grid on; hold off;

% Calc.f(13)
disp(f(27))
```

程序顶层调用结构如图所示，各插值函数根据脚本输入的数据点和参数进行插值，并将结果的 *month-cm* 关系用匿名函数表示，将函数句柄返回，脚本根据曲线函数句柄绘制 *month-cm* 曲线图。



在题中给出的数据是基于月龄分布的，而查询者将通过输入儿童出生日期来查询，因此需要将儿童生日转换为月龄。编写函数 **D2M.m** 来进行转换，当大于 15 天，算一个月。

```
function [n] = D2M(y,m,d)
%DAY calculate the day between the date and today
Y = year(date);
M = month(date);
D = day(date);
n = (Y*12 + M) - (y*12 + m);
k=D-d
if k>15
    n = n + 1;
end
end
```

- **D2M.m**

4.1.2 Newton 插值的分析编写和运行

由于 n 阶多项式插值的唯一性，*Newton* 法和 *Lagrange* 法的最终结果在代数上是一样的，所以仅使用 *Newton* 法进行编程插值。

Newton 插值法的基本公式为

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

其中各项差商可以用差商表的最上面一行构造，如表 2（为适应 *Matlab* 的下标体系，采用从 1 开始计数）。

差商表

Xk	函数值 A(:，1)	一阶差商 A(:2)	二阶差商 A(:3)	三阶差商 A(:4)
X1	**f(x1)**	**f[x1,x2]**	**f[x1,x2,x3]**	**f[x1,x2,x3,x4]**
X2	f(x2)	f[x2,x3]	f[x2,x3,x4]	
X3	f(x3)	f[x3,x4]		
X4	f(x4)			

计算差商表时，可以先生成第一列（函数值），然后使用循环，根据上一列的值进行向量相减相除得到下一列，最后取出第一行 A(1,:)代入公式 3-1 计算即可。

再观察公式可以发现每一项都有重复乘上的公因式，可以进行变形简化减小计算量。

$$\begin{aligned} N_n(x) = f(x_0) + (x - x_0)(f[x_0, x_1] + \ldots \\ + (x - x_{n-2})(f[x_0, \ldots, x_{n-1}] \\ + (x - x_{n-1}) f[x_0, \ldots x_{n-1}, x_n])\ldots) \end{aligned}$$

亦可用逆推实现

$$N_n^{(n)}(x) = f[x_0, \ldots x_{n-1}, x_n]$$
$$N_n^{(k)}(x) = f[x_0, \ldots, x_k] + (x - x_k) N_n^{(k+1)}(x)$$
$$N_n(x) = N_n^{(0)}(x)$$

具体实现时，需要输出两个变量，一个是多项式系数向量，另一个是多项式的计算函数句柄。句柄的输出是为了同一不同插值方法输出的不同函数类型。具体方法是将多项式用 *syms* 类型表示，然后使用内置 *sym2poly* 函数进行简化，得到系数向量，然后使用匿名函数引用 *polyval* 根据多项式系数求值。

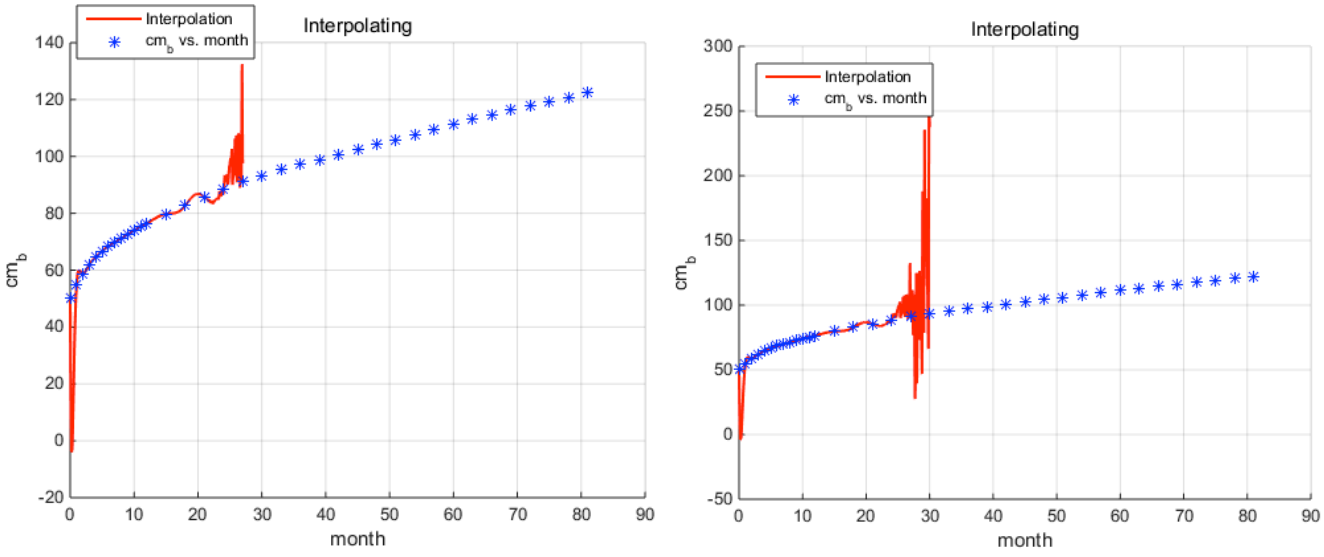
编写 *Newton* 法插值函数（***Newton_itp.m***）如下，程序分为求差商表、产生多项式、化简多项式并获取句柄三步。

- **Newton_itp.m**

```
function [ fX,coefs] = Newton_itp( x,f )
%NEWTON_ITP interpolate 1Ò»D data by Newton Method
%   x-1 x n; f - 1 x n
%   fX - handle of polynomial function
%   coef - the coefficients of polynomial
n = size(x,2);
A = zeros(n,n);
A(:,1) = f'; x=x';
for k=2:n   %calc.Divided difference table
    A(1:n+1-k,k)=(A(1:n+1-k,k-1)-A(2:n+2-k,k-1))...
           ./(x(1:n+1-k)   -x(k:n));
end
syms t;
h=A(1,n);       %produce polynomial
for k=n-1:-1:1
    h=A(1,k)+(t-x(k))*h;
end
coefs=sym2poly(h);
fX=@(t)polyval(coefs,t);    %return the handle
end
```

调用函数时只需输入 cm 和 month 的行向量即可，可以输出函数句柄和多项式参数（从高次到低次），也可只用一个输出变量，仅获得函数句柄。函数句柄可以输入向量作为 x 并输出插值曲线上的 y 向量。

使用脚本 *Main_itp.m* 调用方法，输出得到 Newton 插值曲线图如图所示，



可以看到，在 20 月之前，与趋势比较符合，但是在 20 个月之后，结果不稳定，形成了

很大的峰值，整体趋势不符合预期。另外，在放大区间可以看到，会产生负值。

这是由于阶数较高，产生严重的**龙格现象**。

另外，由于数据点的个数已经给定，所以误差不易计算，且估计误差可能大于实际误差，故难以给出误差，更**无法得到预测置信区间**，不再计算误差。

4.1.3 分段低阶插值法的分析编写与运行

整体的多项式拟合上所述出现了龙格现象。为了解决这个问题，首先尝试使用分段低阶插值。分段低阶插值其实本质上就是分段选择部分数据点，逐段进行 *Newton* 插值或 *Lagrange* 插值。

使用 Matlab 实现分段插值时的最严重的问题在于分段插值得到的是多个多项式组成的分段函数，需要输出分段函数的句柄。实现的方法是使用 *sym2poly*，*mkpp*, *ppval* 等函数。

编写分段低阶插值函数((***Piecewise_itp. m***)如下，程序输入数据点(x,f)和每段的阶数 ord，输出分段函数句柄尽和每一段多项式的参数矩阵 coefs。

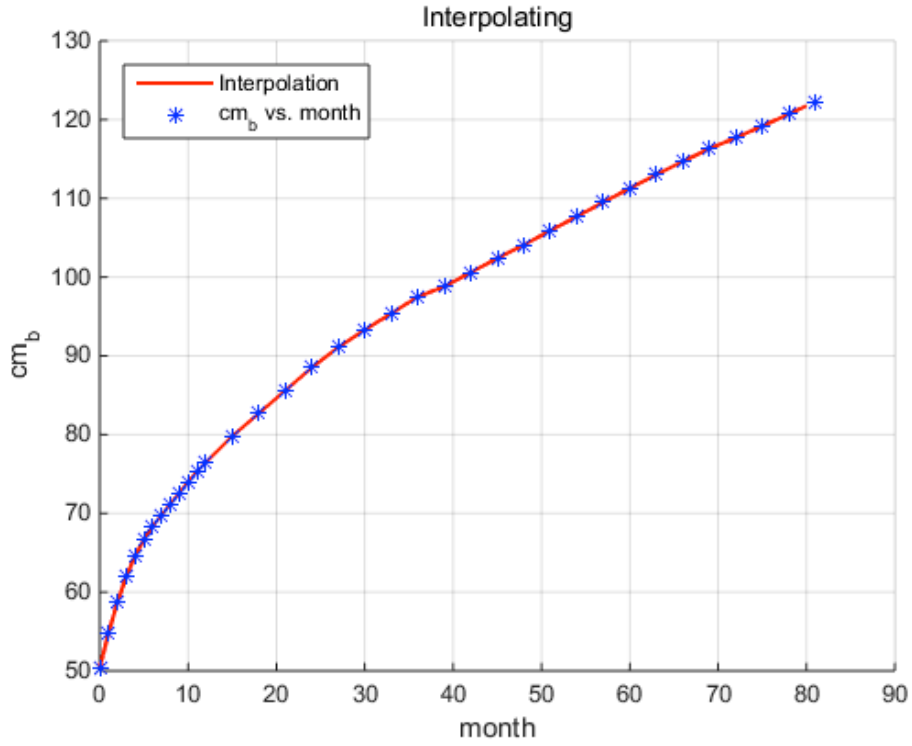
- **Piecewise_itp.m**

```
function [ fX, coefs ] = Piecewise_itp( x, f, ord )
%Piecewise_itp interpolate 1-D data by Piecewise low order polynomial
%   x - 1 x n ; f - 1 x n ; ord - order of polynomial
%   fX - handle of polynomial function
%   coefs - the coefficients of polynomials
n=size(x,2);
syms t;
sxf=sortrows([x' f'],1);           % sort by x
x=sxf(:,1)';f=sxf(:,2)';

breaks=[];coefs=[];
for i=1:ord:n-1
    x_t=x(i:min(n,i+ord))-x(i);
    f_t=f(i:min(n,i+ord));
    [~,coef_t]=Newton_itp(x_t,f_t); % low order polynomial
    coefs(end+1,:)=coef_t;
    breaks(end+1)=x(i);
end

breaks(end+1)=x(end);
pp=mkpp(breaks,coefs);             % make piecewise polynomial
function
fX=@(t)ppval(pp,t);               % return the handle
end
```

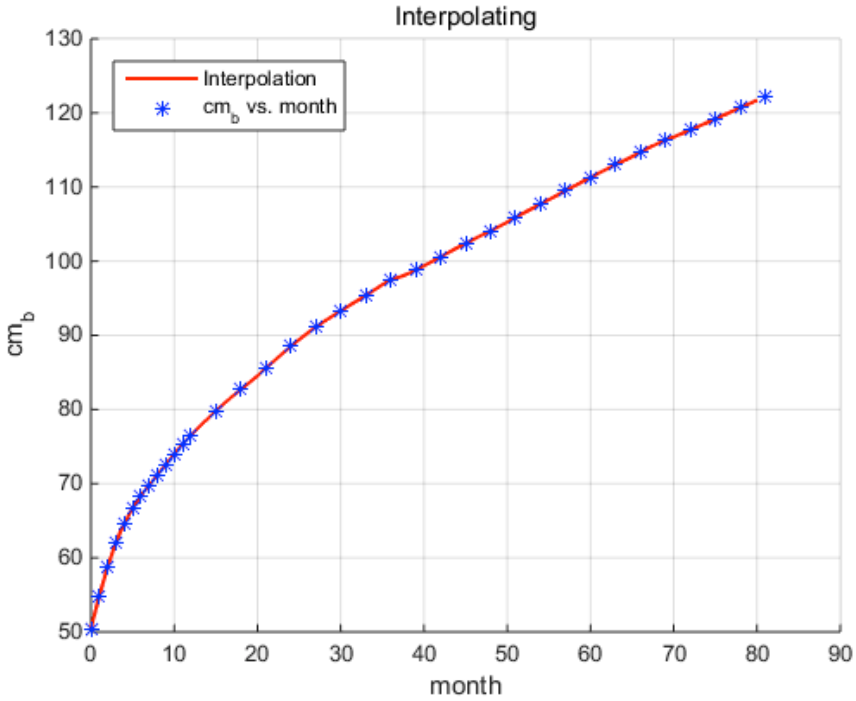
由于分段需要按照自变量升序排列，所以先使用 sortrows 函数按 x 排序，然后分段调用 Newton 插值，得到输出的多项式的各次系数。将每一段的系数和分断点排列成矩阵，使用

mkpp 生成分段函数结构体，然后使用 ppval 函数产生函数句柄。

为了观察阶数对插值结果的影响，在脚本中调用阶数为 1,5,3,35 的分段低阶插值，绘制结果曲线（见下两页)。

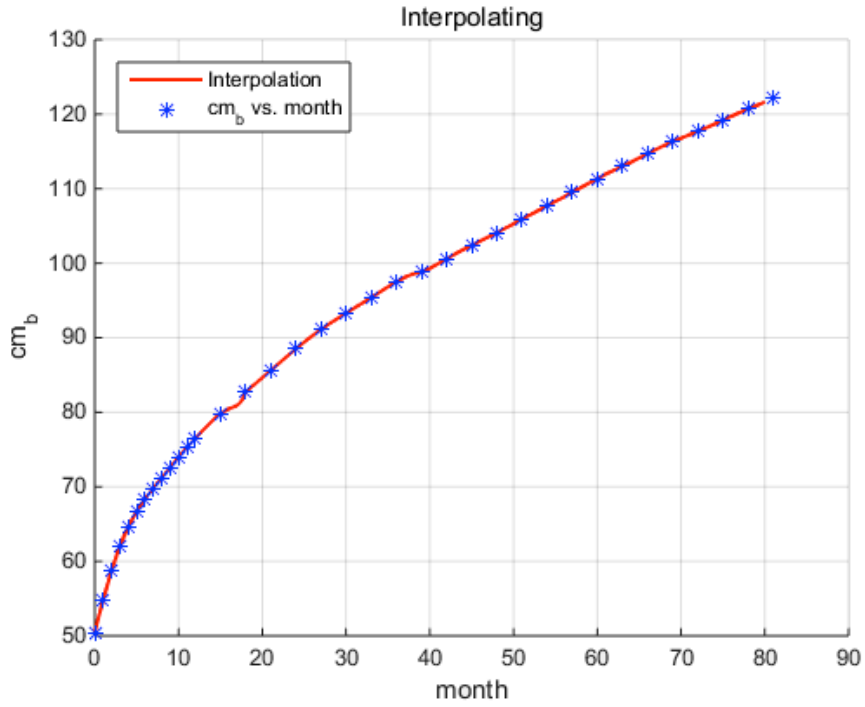


分段低阶插值结果（1 阶 x35 段）

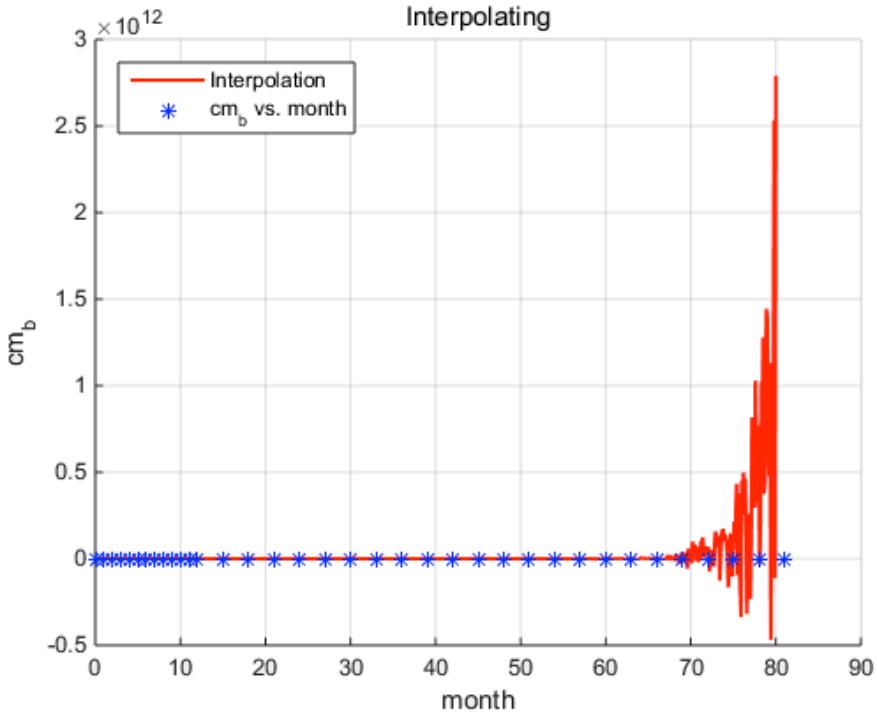


分段低阶插值结果（5 阶 x7 段）

可以看到，一阶插值时函数趋势完全符合预期，但是**间断点光滑性差**。但远远好于高阶 *Newton* 插值，另外，五阶插值在分段点处也不光滑。



分段低阶插值结果（7 阶 x5 段）



分段低阶插值结果（35 阶 x1 段）

可以看到，阶数达到 7 时，与 1、5 阶比较类似，阶数达到 35 时的结果其实就是 *Newton* 法的结果，并没有分出段来。

比较各阶数的结果可以看到，各结果在数据点较为整齐的 month 较小处都比较平滑，但是随着阶数增加，光滑性变好的同时，稳定性变差。

4.1.4 三阶样条插值法的分析编写与运行

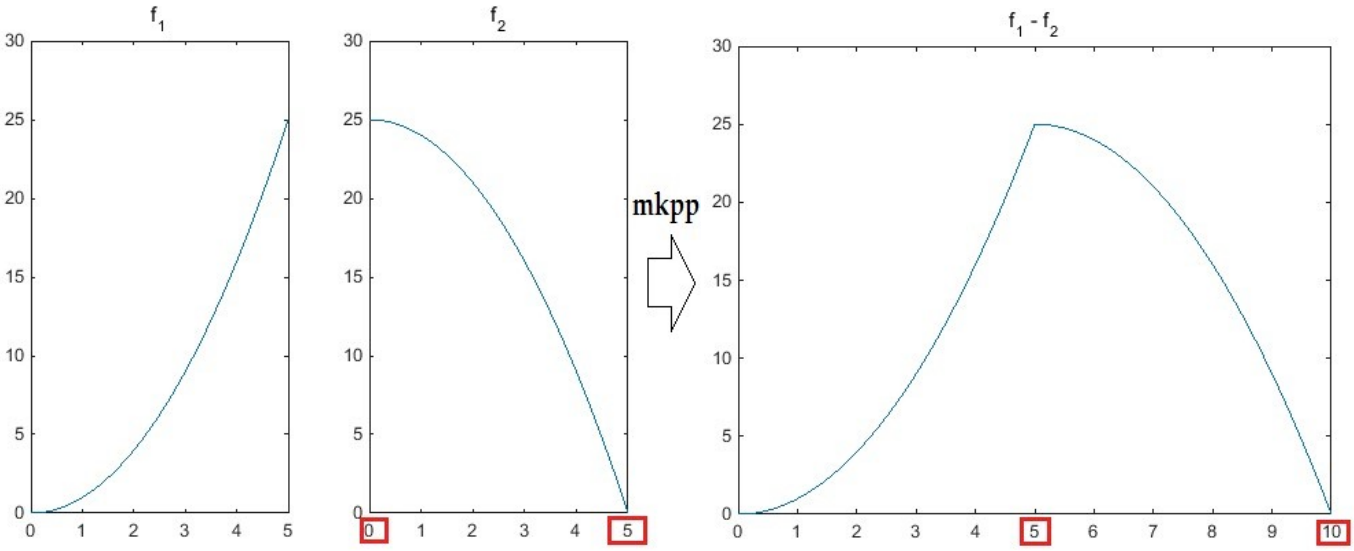
为了解决各段光滑性和稳定性的问题，使用三弯矩法编写三阶样条插值程序。并且选择自然边界条件。

具体过程主要分为生成矩阵、求解线性方程组、根据 Mi 得到最后的每一段的方程。具体求法此处不再赘述。但是需要提出以下三点编程时的注意点:

- 由于 *Matlab* 的下标是从 1 开始的，所以需要将公式的下标进行小心的更改，编程时应格外注意前后下标是否一致对应;
- 生成的矩阵是对角占优的三对角矩阵，所以可以利用 *Thomas* 算法进行快速求解;
- 进行向量逐项乘除计算时注意要用“./”运算符而不是‘./”。

另外，类似于 4.1.3，使用 *mkpp* 等函数输出分段多项式的形式。

➢ 关于 mkpp 函数，有一点要特别注意(4.1.3 中未说明)，使用时需格外注意:mkpp 函数输入的各段多项式系数，转换后会把 0 移动到该段起始点。即如下图所示，编程时需要注意这点，在生成函数时将函数向左平移 x0。



上图为 mkpp 函数特性说明（仅仅是个例子，与本题无关，注意横坐标）

- **CubicN_itp.m**

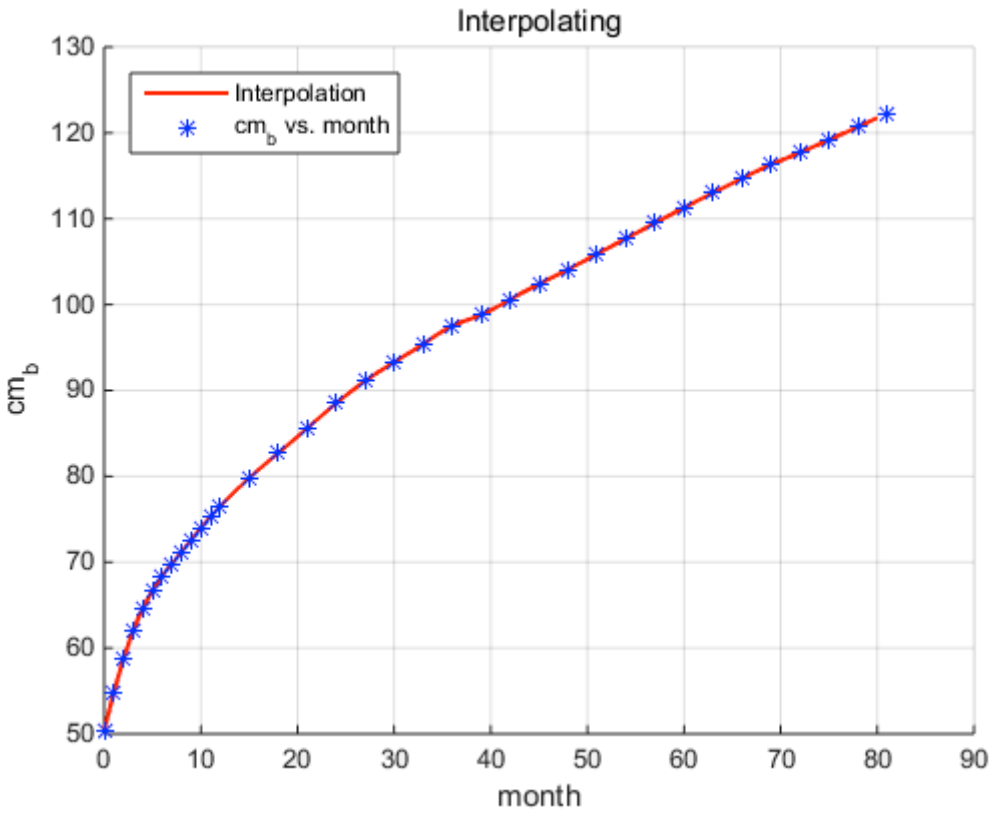
```
function [ fX ] = CubicN_itp( x, f )
%CubicN_itp interpolate 1-D data by Cubic spline, with nature bound.
%   x - 1 x n ; f - 1 x n ;
%   fX - handle of polynomial function
n=size(x,2);
sxf=sortrows([x' f'],1);        % sort by x
x=sxf(:,1)';f=sxf(:,2)';
% Calc Df, Dx, mu, lambda, g
Dx=diff(x);
Df=diff(f);
Df=Df./Dx;
mu(1:n-2)=1./(1+Dx(2:n-1)./Dx(1:n-2));
lambda=1-mu;
g(1:n-2)=(Df(2:n-1)-Df(1:n-2))./(Dx(2:n-1)+Dx(1:n-2))*6;
% Make matrix
A=diag(ones(1,n-2)*2,0)+diag(lambda(1:n-3),1)+diag(mu(2:n-2),-1);
% Solve M
M=LU_Tm_solve(A,g')';
M=[0,M,0];

syms t s;coefs=[];
for i=1:n-1
    t=s+x(i);                % notice to move left bound to 0 (mkpp())
    h=(M(i)*(x(i+1)-t)^3+M(i+1)*(t-x(i))^3)/(6*Dx(i)) ...
        +(f(i)  -M(i)  *Dx(i)^2/6)*(x(i+1)-t)/Dx(i) ...
        +(f(i+1)-M(i+1)*Dx(i)^2/6)*(t-x(i))/Dx(i); % one of spline
poly.s
    coefs(end+1,:)=sym2poly(h);
end
pp=mkpp(x,coefs);            % make piecewise polynomial function
fX=@(t)ppval(pp,t);          % return the handle
end
```

- **LU_Tm_solve.m**

```
function [x] = LU_Tm_solve( A, b, ~)
%LU_Tm_solve Solve linear eqs by LU decomposition - Thomas
%   A: n x n; b: n x m; x: n x m;
n=size(b,1);
aa=diag(A,-1);bb=diag(A);cc=diag(A,1);
% Thomas decp.
L=diag(aa,-1);U=diag(ones(1,n));
L(1,1)=bb(1);
for k=2:n
    U(k-1,k)=cc(k-1)/L(k-1,k-1);
    L(k,k)=bb(k)-aa(k-1)*U(k-1,k);
    % vpa([L U],4) % display LU
end
% Solve Y
y=zeros(size(b));
y(1,:)=b(1,:)/bb(1);
for k=2:n
    y(k,:)=(b(k,:)-aa(k-1)*y(k-1,:))/(bb(k)-aa(k-1)*U(k-1,k));
```

```
end
% Solve X
x=zeros(size(b));
x(n,:)=y(n,:);
for k=n-1:-1:1
    x(k,:)=y(k,:)-U(k,k+1)*x(k+1,:);
end
end
```

在脚本中调用三次样条，绘制插值结果曲线如图所示



三次样条插值结果

可以看到插值结果曲线在转折点处光滑程度很好，且整体趋势良好，未出现下降区间。但是 p 接近 40 时斜率有点不正常，这是由于数据误差，采用插值必然有这问题。

4.2 拟合方法的编写和分析

4.2.1 多项式最小二乘法拟合的编写

现考虑采用拟合的方法。由于数据关系比较简单，本文中使用多项式函数，采用最小二乘法进行拟合。

最小二乘法的实现可以用矩阵线性代数方程求解完成

$$A\begin{bmatrix} a_0 \\ \\ a_n \end{bmatrix} = b;\ \ A = C^T C,\ \ b = C^T Y;\ \ C = \begin{bmatrix} x_1^0 & & x_1^n \\ \\ x_m^0 & & x_m^n \end{bmatrix},\ \ Y = \begin{bmatrix} y_1 \\ \\ y_m \end{bmatrix}$$

最终得到的回归方程为

$$\hat{y} = a_0 + a_1 x + a_2 x^2 + \quad a_n x^n$$

公式中 A 为对称阵，可以尝试 *Cholesky* 以加快计算速度，当阶数不高，可直接使用高斯法求解。此处的阶数并不算高。采用高斯法也可以立刻得到结果。

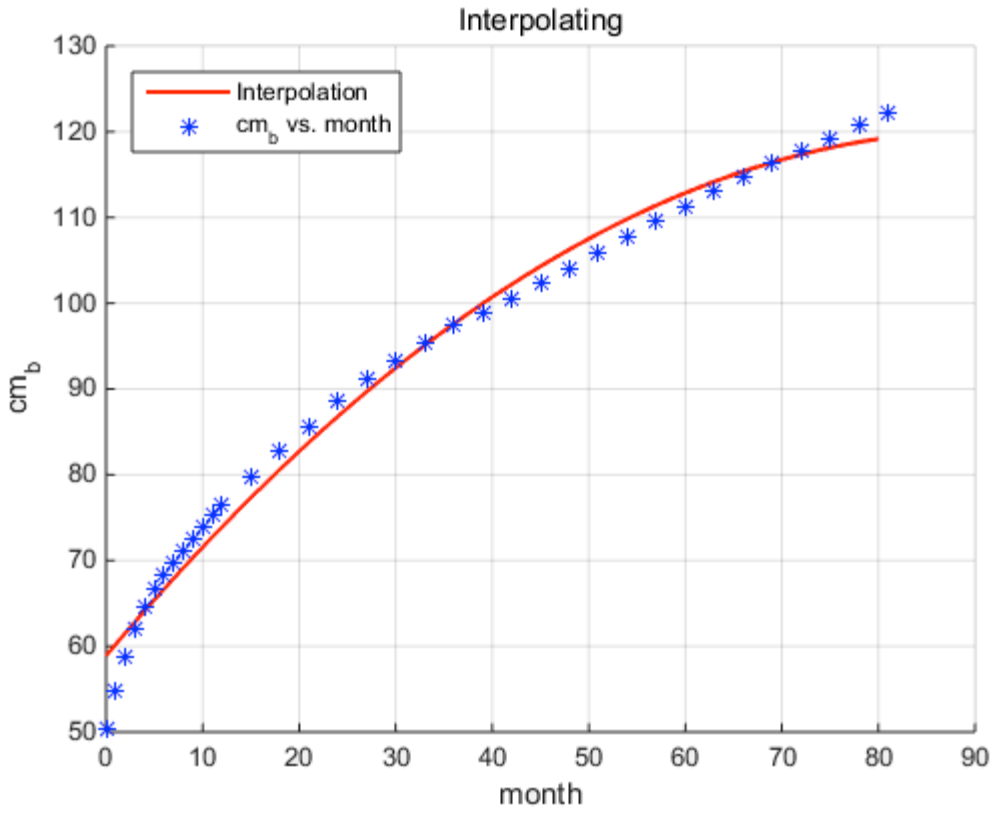
- **Polynomial_fit.m**

```
function [ fX,coefs ] = Polynomial_fit( x, f, degree )
%Polynomial_fit fitting 1-D data with polynomial
%   x - 1 x n ; f - 1 x n ; degree - the degree of polynomial
%   fX - handle of polynomial function
%   coefs - the coefficients of polynomial
n=size(x,2);

Y=f'; c=x';                     % calc. Matrix
C=ones(n,degree+1);
for k=1:degree
    C(:,k+1)=C(:,k).*c;
end
A=C'*C; b=C'*Y;
disp(['Degree: ' num2str(degree) '; Condition number: '
num2str(cond(A))]);
coefs=Gauss_solve(A,b,[1.1,1e-7]); % solve coefs of polynomial
coefs=fliplr(coefs');            % return coefficients
fX=@(t)polyval(coefs,t);          % return the handle
end
```

➢ **Gauss_solve.m**

```
function [x] = Gauss_solve( A, b, ~)
%Gauss_solve Solve linear eqs by Gauss elimination Method
%   A: n x n; b: n x 1; x: n x 1;
n=size(b,1);
% forward elimination
for k=1:n-1
    factor=A(k+1:n,k)/A(k,k);
    A(k+1:n,:)=A(k+1:n,:)-factor*A(k,:);
    b(k+1:n  )=b(k+1:n  )-factor*b(k  );
    %vpa([A b],6) % display Ab
end
% back substitution
x=zeros(n,1);
x(n)=b(n)/A(n,n);
for i=n-1:-1:1
    x(i)=(b(i)-A(i,:)*x(:))/A(i,i);
end
end
```

得到拟合曲线如下：



函数输入数据点和所需多项式次数，使用上次作业编写的高斯法程序求解，输出拟合函数句柄和拟合多项式的各系数向量。注意由于方程解出的系数是低次项到高次项的列向量，而 polyval 函数是需要高次项到低次项的行向量，所以要先转置再用 fliplr 左右翻转。另

外，顺便计算输出了矩阵 A 的条件数。

4.2.2 均方误差等参数的计算及绘图脚本的编写

编写脚本调用多项式拟合函数，绘制拟合函数图和残差图，并按下式计算均方误差。

$$MSE = \frac{1}{n}\sum (y - y)^2$$

编写脚本 ***Main_fit. m*** 进行各次数的拟合，代码如下

- **Main_fit.m**

```
%Main_fit.m fitting, calc. MSE, and plotting
% Given data
load('month.mat');
load('cm_b.mat');
% Fitting in diff. degree
for deg=1:6
    [f,coefs]=Polynomial_fit(month,cm_b,deg); % use 1-35-degree
fitting

% Plotting
    x=0:0.1:80;y=f(x);
    figure;subplot(2,1,1);hold on;
    plot(x,y,'r','linewid',1.5);
    plot(month,cm_b,'b*');
    legend(gca, 'Fitting', 'cm_b vs. month', 'Location',
'NorthWest' );
    xlabel 'month'; ylabel 'cm_b';
    title([num2str(deg) '-Degree Fitting']);
    grid on; hold off;

% Residuals Diagram
    subplot(2,1,2);
    stem(month,f(month)-cm_b);
    xlabel 'month'; ylabel 'Residual';
    MSE=sum((f(month)-cm_b).^2)/length(month);     %
Root-mean-square error
    title (['  Residuals Diagram (MSE=' num2str(MSE) ')']);
    axis([0 90 -3 3]);
    grid on;
    disp([' Coefficients: ' num2str(coefs)]);
    disp([' MSE: ' num2str(MSE)]);
    print(gcf, '-dbmp', ['deg' num2str(deg) '_fit']);
end
```

4.2.3 拟合结果及分析

运行脚本，工作区输出文本如下：

```
>> Main_fit
Degree: 1; Condition number: 4571.8385
   Coefficients: 0.788159          64.2082
   MSE: 20.1983
Degree: 2; Condition number: 32522055.7187
   Coefficients: -0.00730169          1.33743          58.9011
   MSE: 5.8549
Degree: 3; Condition number: 259922268675.4294
   Coefficients: 0.000184399      -0.0291362          1.99589          55.5625
   MSE: 2.0037
Degree: 4; Condition number: 2050284234966830
   Coefficients: -4.85145e-06    0.000954877      -0.0676283          2.63004            53.498
   MSE: 0.78172
Degree: 5; Condition number: 1.525052174194864e+19
   Coefficients:   1.16362e-07  -2.79532e-05       0.00255878        -0.113193            3.09323
52.497
   MSE: 0.47773
Degree: 6; Condition number: 1.44634346455427e+22
   Coefficients:  -4.68029e-09       1.2384e-06  -0.000128727       0.00672816        -0.191278
3.63657           51.6992
   MSE: 0.27906
```

整理输出结果，汇总拟合结果如下表:

拟合结果各项参数

次数 degree	x6	x5	X4	X3	X2	X	1
1	--	--	--	--	--	0.788159	64.2082
2	--	--	--	--	-0.0073017	1.33743	58.9011
3	--	--	--	0.0001844	-0.0291362	1.99589	55.5625
4	--	--	-4.85E-06	0.00095488	-0.0676283	2.63004	53.498
5	--	1.16E-07	-2.80E-05	0.00255878	-0.113193	3.09323	52.497

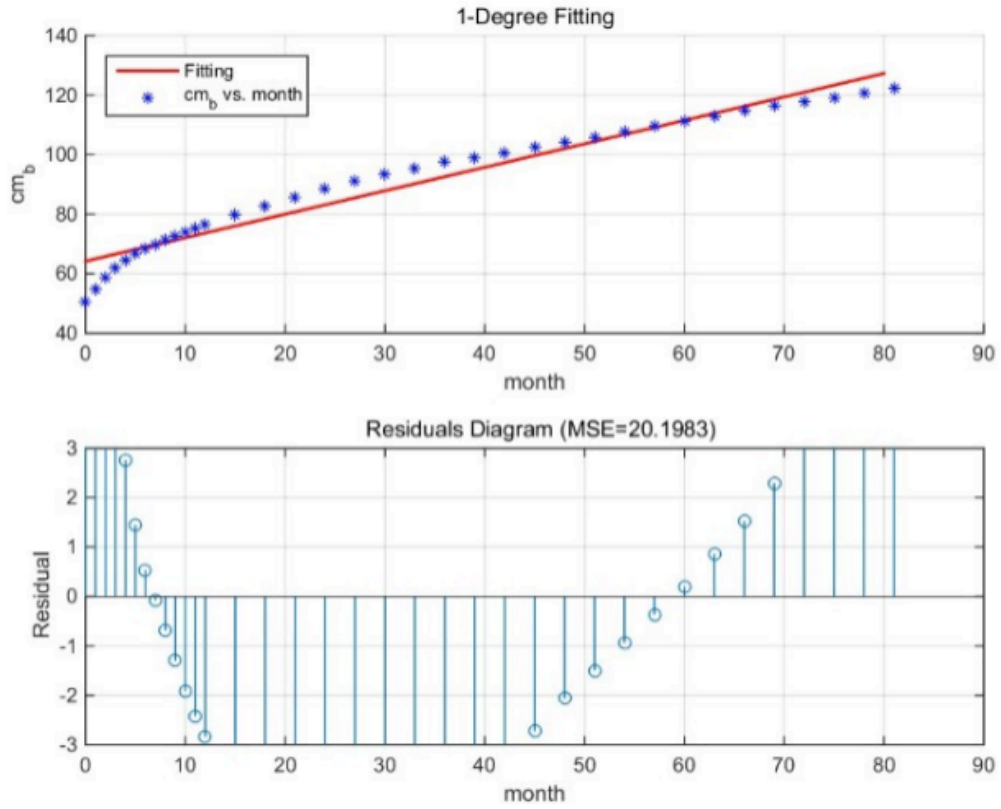
6	-4.68E-09	1.24E-06	-0.0001287	0.00672816	-0.191278	-0.191278	51.6992

下图为拟合结果性能指标等参数

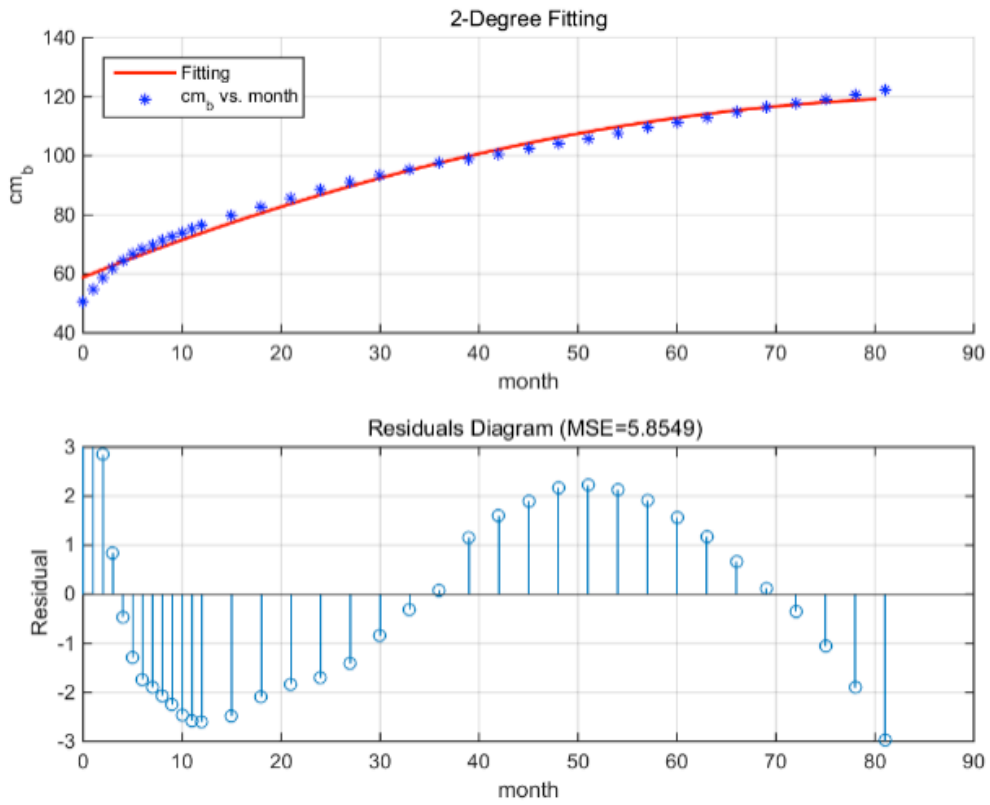
拟合结果性能指标等参数

次数 degree	A 的条件数	均方误差 MSE
1	4571.84	20.1983
2	32522055.72	5.8549
3	259922268675.43	2.0037
4	2050284234966830.00	0.7817
5	15250521741948600000.00	0.4777
6	144634346455427000000000.00	0.2791

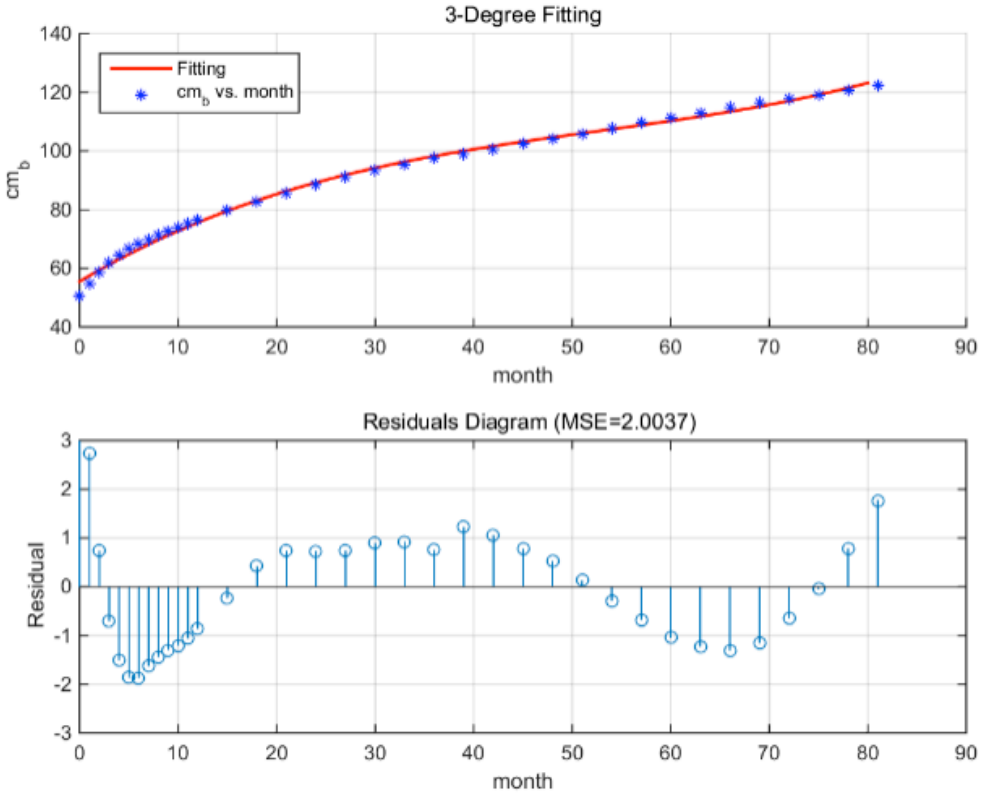
输出的拟合曲线图和残差图如下 6 张图所示（见下三页）。



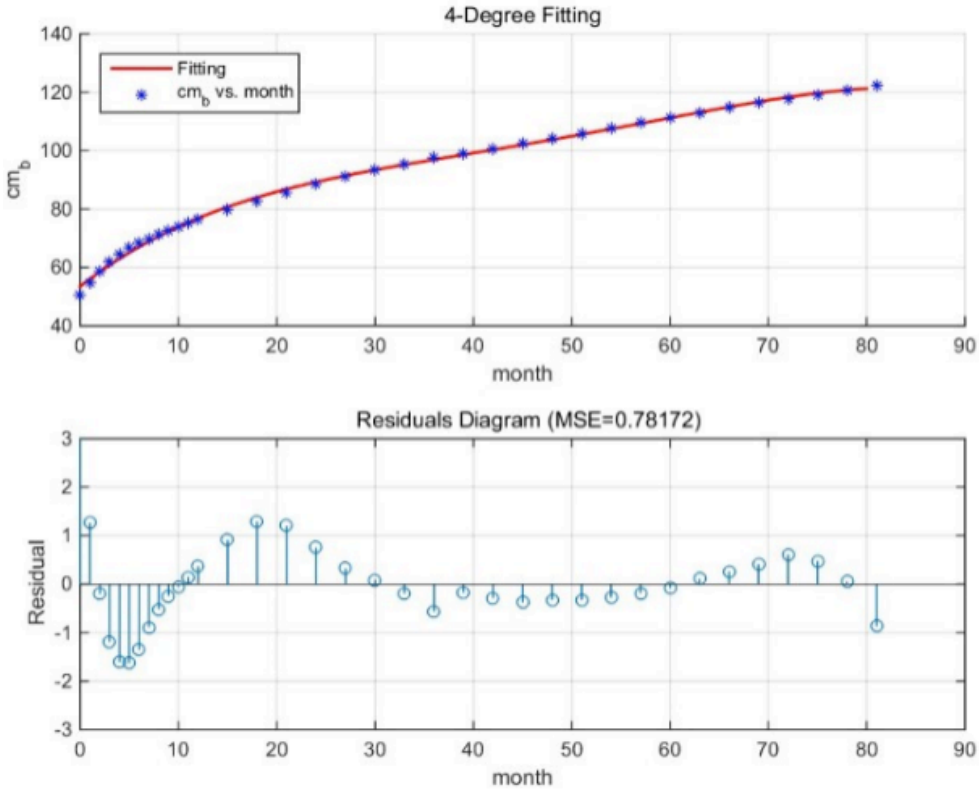
1 次多项式拟合函数曲线图和残差图



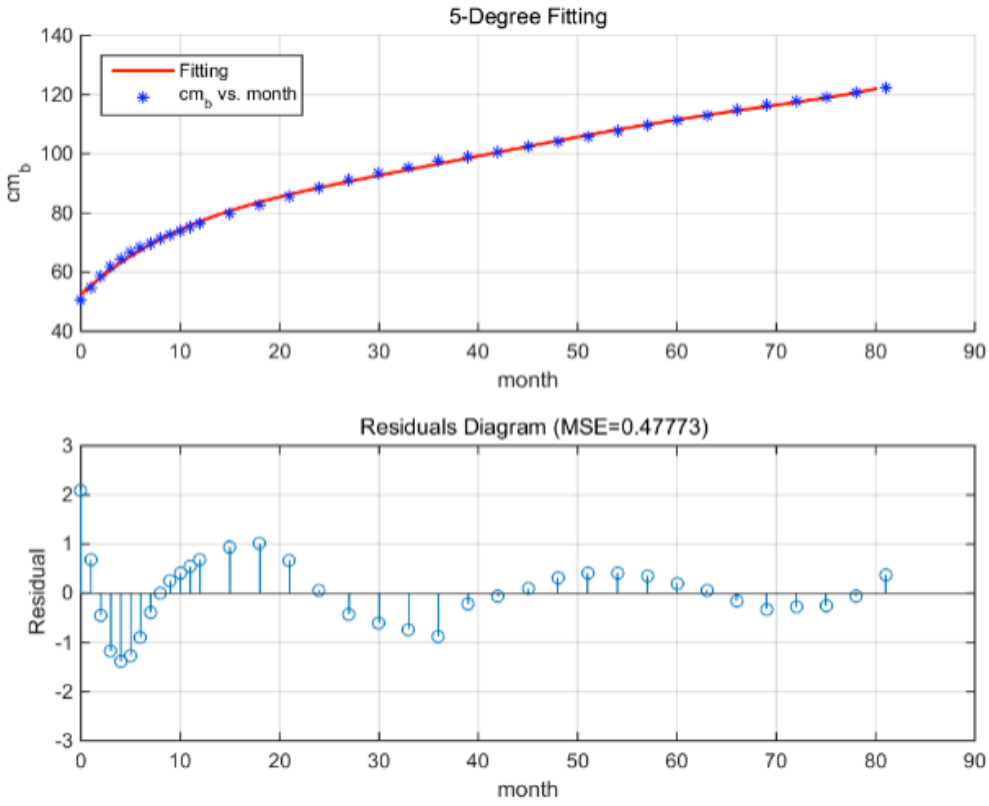
2 次多项式拟合函数曲线图和残差图



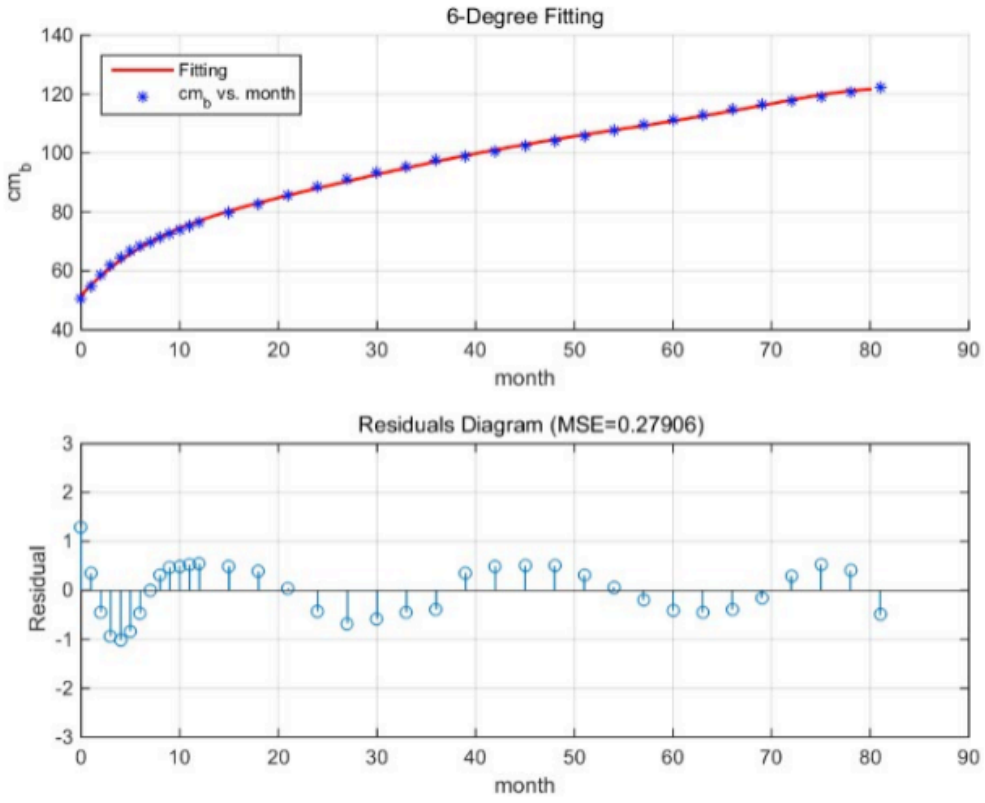
3 次多项式拟合函数曲线图和残差图



4 次多项式拟合函数曲线图和残差图



5 次多项式拟合函数曲线图和残差图



6 次多项式拟合函数曲线图和残差图

次数较低时，拟合结果的稳定性等等都较好，数据点均匀分布在预测点两侧，但是残差稍大。次数增加，均方误差减小。次数达到三次时，三次项系数很小，结果与二次拟合很接近，但残差有减小。达到 6 次的时候 每个点基本上都吻合，误差仅来自于舍入误差。

观察表 5 可以看到，随着次数的增加，矩阵条件数急剧增加，致使方程为病态，最后结果曲线不稳定。而随着次数增加，无论方程是否病态，均方误差都是在减小的。

4.2.4 参数和预测置信区间的计算程序及结果

使用回归分析时，可以得到拟合参数的置信区间和预测函数的置信区间。根据数理统计相关知识不难推出，参数 a;的置信水平为 a 的置信区间为

$$\left(a_i \pm t_{\frac{\alpha}{2}}(n-k-1)\cdot\sqrt{c_{ii}\cdot\frac{MSE}{n-k-1}} \right)$$

式中，k 为次数，$c_{ii}=\left[\left(X^TX\right)^{-1}\right]_{ii}$

预测值 y 的预测区间为

$$\left(f(x) \pm t_{\frac{\alpha}{2}}(n-k-1)\cdot\sqrt{(1+X*CX*^T)\cdot\frac{MSE}{n-k-1}} \right)$$

式中，$X*=(1,x,x^2,...,x^k)^T$

根据上述两式，在 4.2.2 的脚本的基础上进行改进，得到可以计算参数置信区间和预测曲线的程序。编写脚本如下

- **Main_fit_Pro.m**

```
%Main_fit_Pro.m fitting, calc. confidence interval
% Given data
load('month.mat');
load('cm_b.mat');
% Fitting in diff. degree
deg=1;
[f,coefs]=Polynomial_fit(month,cm_b,deg); % use 1-6-degree fitting

% Plotting
x=0:0.25:70;y=f(x);
figure;hold on;
plot(x,y,'r','linewid',1.5);
plot(month,cm_b,'b*');
xlabel 'month'; ylabel 'cm_b';
title([num2str(deg) '-Degree Fitting']);
grid on;

alpha=0.05;n=size(month,2);
X=ones(n,deg+1);
for k=1:deg
    X(:,k+1)=X(:,k).*month';
end
MSE=sum((f(month)-cm_b).^2)/length(month);     % Root-mean-square
error
delta_coefs=tinv(1-alpha/2,n-deg-1)*sqrt( ...
            MSE/(n-deg-1)*diag(inv(X'*X)))';
syms t;h_1=[];h_2=[];
for k=0:deg
    h_1=[h_1;t.^k];
    h_2=[h_2,t.^k];
end
delta_Y(t)=tinv(1-alpha/2,n-deg-1)*sqrt( ...
            MSE/(n-deg-1).*(1+h_2*(inv(X'*X)*h_1)));
delta_y=eval(delta_Y(x));
plot(x,y+delta_y,'b-.',x,y-delta_y,'b-.');
legend(gca, 'Fitting', 'cm_b vs. month','Prediction bound', ...
    'Location', 'NorthWest' );
disp('  Coefficients:');
disp([coefs-delta_coefs;coefs+delta_coefs]);
print(gcf, '-dbmp', ['deg' num2str(deg) '_fit_p']);
```

```
>> Main_fit_Pro
Degree: 1; Condition number: 4571.8385
   Coefficients:
     0.365511253335685    64.198097814683351
     1.210806917723722    64.218344457755009

>> Main_fit_Pro
Degree: 2; Condition number: 32522055.7187
   Coefficients:
   -0.313122340317302      1.315986235085317    58.900858203708609
     0.298518960422641      1.358870339669894    58.901408935993999

>> Main_fit_Pro
Degree: 3; Condition number: 259922268675.4294
   Coefficients:
   -0.232137405303369              -0.060363051401079                    1.994927777240848
55.562525671842643
     0.232506202326008                0.002090578862007                  1.996846323829578
55.562541636568035
```

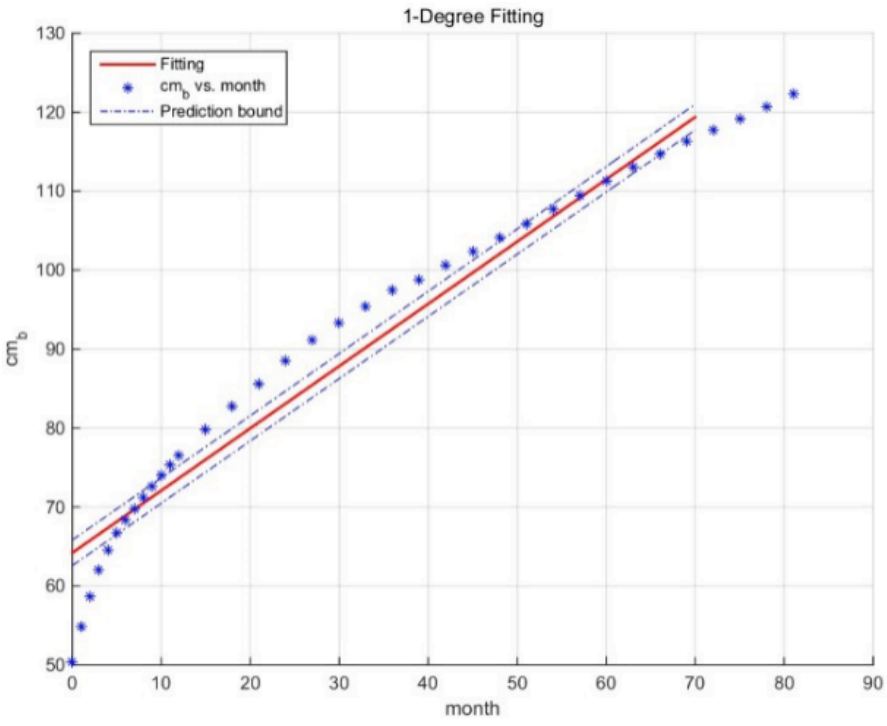
令次数 deg=1，2，3 时，输出结果如下

整理得参数置信区间和预测区间如下：

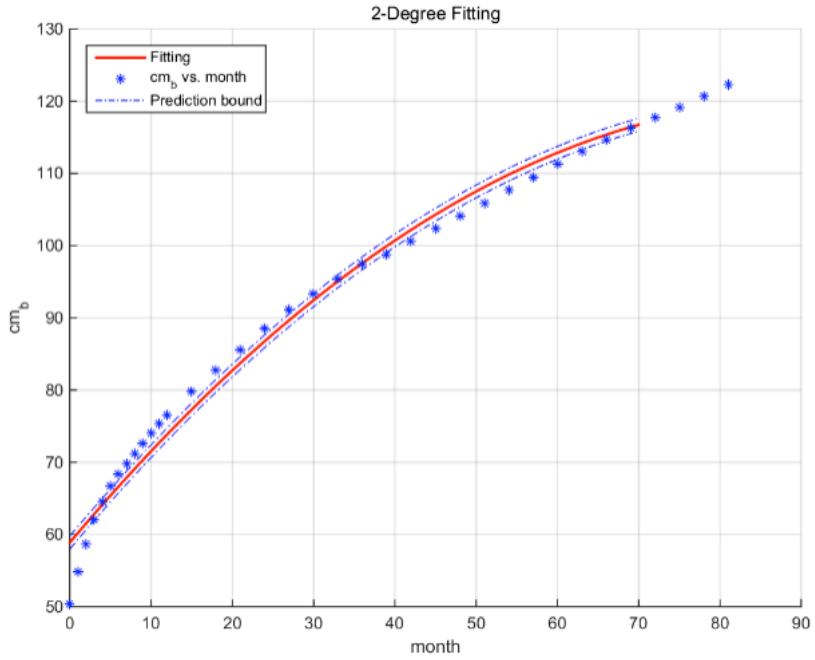
1，2，3 次拟合下的参数置信区间和预测区间

次数 degree	参数置信区间			
	x3	x2	x1	1
1	-	-	(0..365, 1.210)	(64.198,64.218)
2	-	(-0.313,0.298)	(1.315,1.358)	(58.9008,58.9014)
3	(-0.23,0.23)	(-0.06,0.00)	(1.994,1.996)	(55.56252,55.56254)

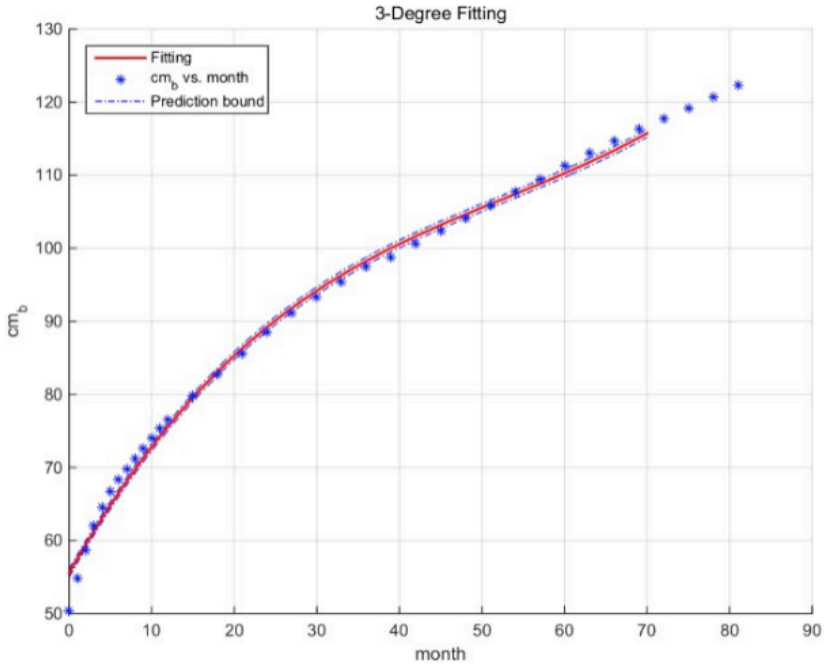
输出图像如下：



1 次拟合下的拟合曲线及预测区间



2 次拟合下的拟合曲线及预测区间



3 次拟合下的拟合曲线及预测区间

因此，综合来看，考虑光滑性稳定性，均方误差等因素，三阶多项式比较适合拟合，拟合结果如下：

回归方程： $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

系数： $a_3 = 0.000184$， $a_2 = -0.0291362$， $a_1 = 1.99589$， $a_0 = 55.5625$

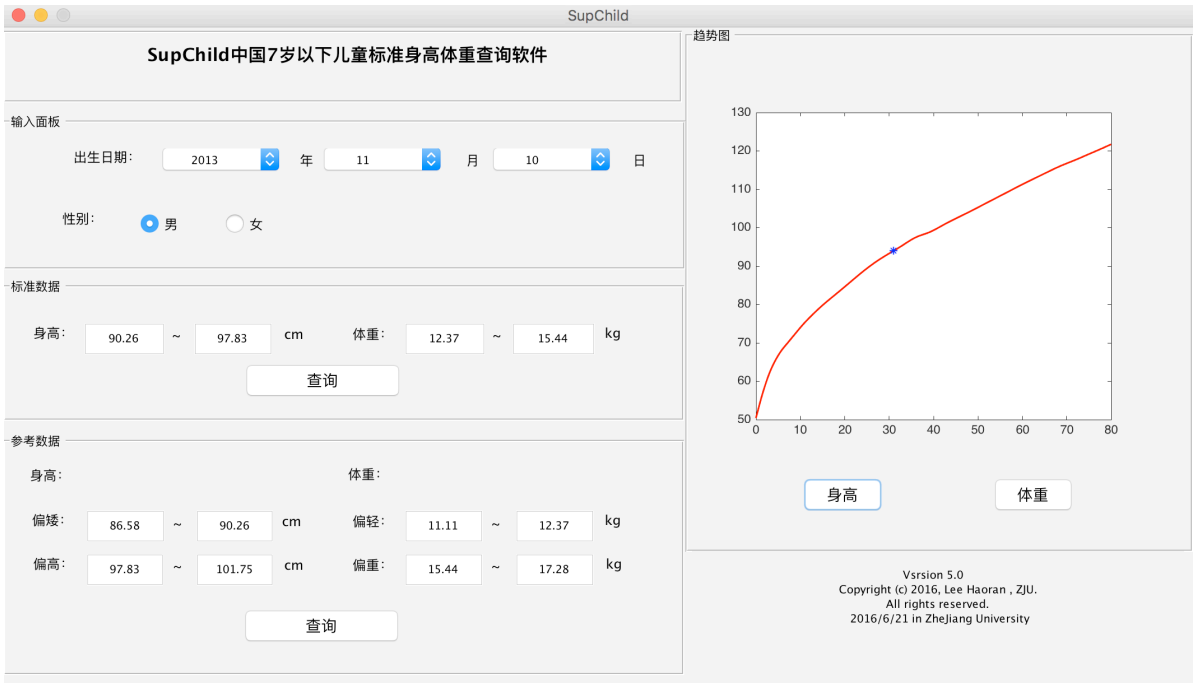
均方误差： $MSE = 2.0037$

5 界面设计

5.1 程序编译及运行环境需求

程序编译的环境为 Mac 系统，Matlab2015b 环境。

5.2 程序界面及前端功能设计



程序主界面

根据设计，程序几乎全部功能都可在主界面实现。界面分为输入面板（上方）、按键、与输出面板（下方）。

详细使用方法请见 **说明文件。**

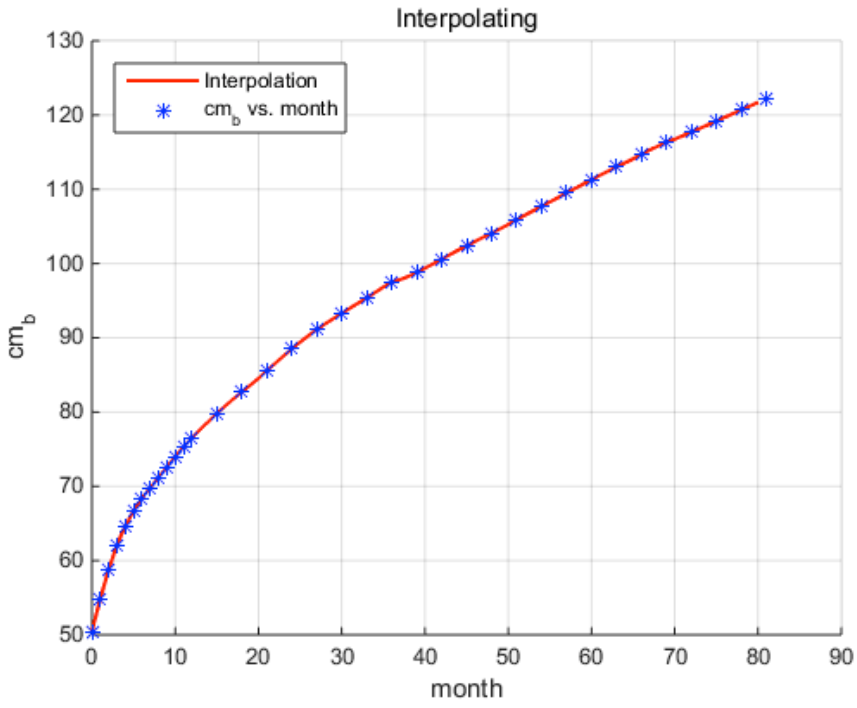
6 性能描述（运行时间，准确性）

6.1 运行时间

本程序最初的设计是利用三阶样条插值法进行编写。(采用插值而不采用拟合的原因前文已述并分析，在此不再赘述)，但是当版本 1.0 完成后，发现利用三阶样条插值的速度较慢，会让使用者感觉有延时，实用性不佳，因此改变了算法。采用分段低阶插值法(7 阶 5 段）来插值。经改变算法之后，应用运行一次的时间为 1s 左右，不会让使用者感到明显的延迟。

6.2 准确性

下图为分段低阶插值法（7 阶 5 段）的曲线图，从图上可以看出曲线光滑性，准确性都良好。为了兼顾运行速度，因此采用了此算法。其他分析见上文算法描述。



7 主要附录

File	Type	Explanation
SupChild.m	Function	The main function of the App
SupChild.fig	GUI	The main part of the App
D2M.m	Function	Change the date
perdict.m	Script	Predict the trend of the line
Gauss_solve.m	Function	Gauss_solve
Newton_itp.m	Function	Newton interpolate
LU_Tm_solve.m	Function	LU_Tm_solve
Polynomial_fit.m	Function	Polynomial_fit.
Polynomial_fit..m	Function	Polynomial_fit.
Main_fit.m	Function	The main function
CubicN_itp.m	Function	CubicN interpolate
Main_itp.m	Function	Main_itp
Main_fit_Pro.m	Function	Main_fit_Pro
cm_()_().mat	Cell	The cell of height
km_()_().mat	Cell	The cell of the weight