第五次作业

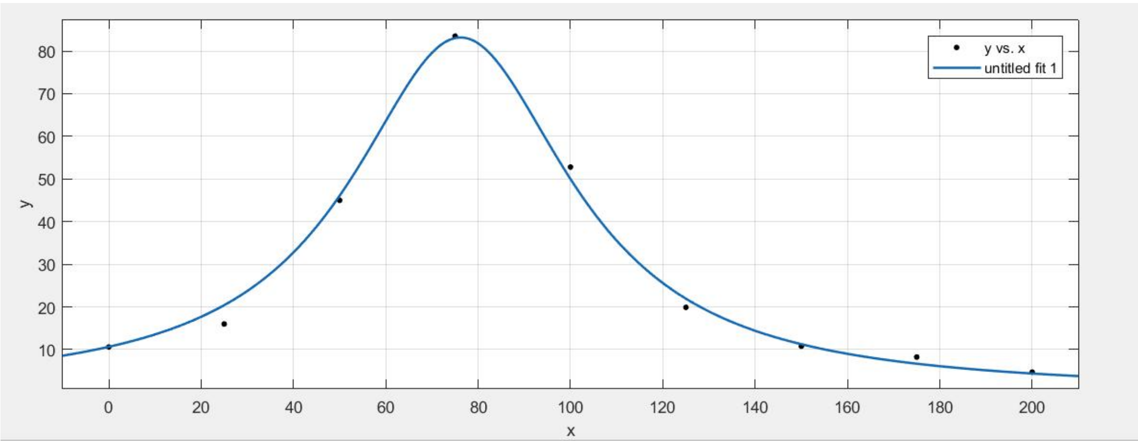
李阳 515072910019 黄弘毅 515072910038 王潇卫 515072910032

1

The theoretical formula of the cross section with neutron energy is $f(E) = \frac{f_r}{[(E-E_r)^2+\tau^2/4]^2}$ These three parameters: E_r means the location of the resonant peak; τ means the full width at half maximum of this resonant process. The deravitives: $f'(E) = \frac{-2f_r(E-E_r)}{[((E-E_r)^2+\tau^2/4)]^2}$ $f''(E) = -2f_r\frac{[((E-E_r)^2+\tau^2/4)]-4(E-E_r)^2}{[(E-E_r)^2+\tau^2/4]^3}$ The boundary condition we choose is based on the value of f'(E) and f''(E) at E=0 and 200Mev. In order to get the E_r and τ from the dataset, we just need to use free boundary conditions,for the boundary conditions have a small effect on the resonant peak and full width at half maximum of this resonant process.

```
clear all;clc;
x = 0:25:200;  y = [10.6 16.0 45.0 83.5 52.8 19.9 10.8 8.25
4.7];%dataset
xx = 0:0.1:200;
yy = spline(x,y,xx);
[Max,Index] = max(yy);
Er = xx(Index);
half = Max/2;
delta = abs(yy-half);
```

We get the result: Er=76.2MeV τ=58.4MeV We get the result: Er=76.2MeV τ=58.4MeV So now the formula is $f(E) = \frac{f_r}{[(E-76.2)^2+852.64]^2}$ In order to determin the parameter f_r, we use the matlab built-in curving fitting:



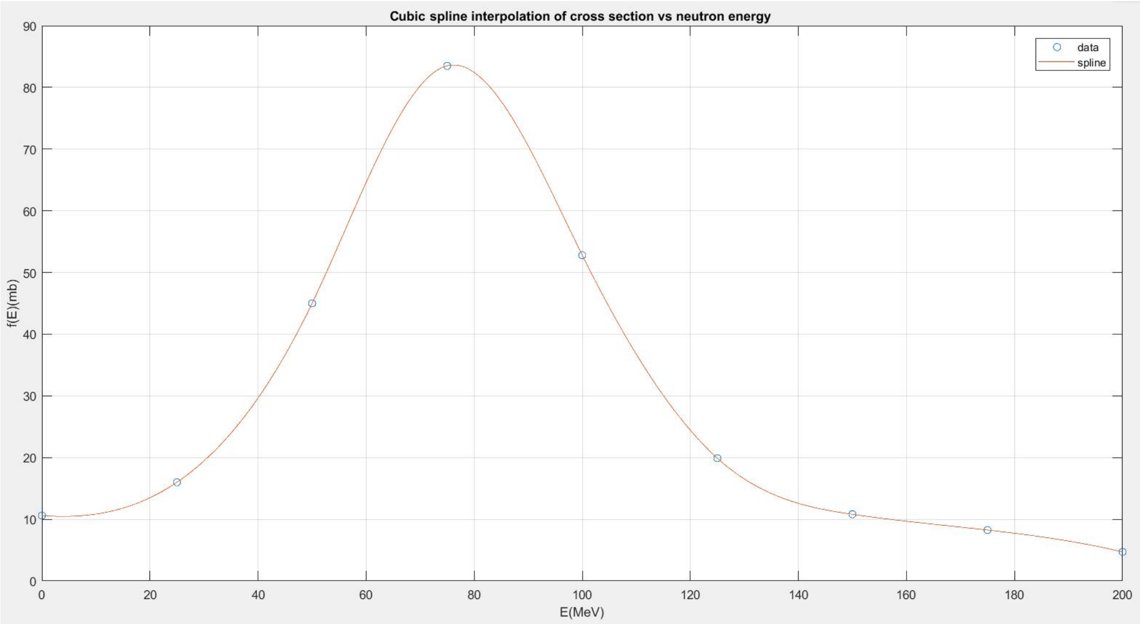
we get $f_r = 7.093e+04$ Now we can estimate the boundary conditon $f''(0) = 0.00795$ $f''(200) = 0.0015$ They are nearly equal to 0, So we can apply the free boundary conditions in this interpolation.

```
%% the script for question 1
clear all;clc;
x = 0:25:200;  y = [10.6 16.0 45.0 83.5 52.8 19.9 10.8 8.25
4.7];%dataset
f1 = spline(x,y,10);
f2 = spline(x,y,90);
f3 = spline(x,y,185);
xx = 0:0.1:200;
yy = spline(x,y,xx);
plot(x,y,'o',xx,yy,'-')
grid on
xlabel('E(MeV)')
ylabel('f(E)(mb)')
title('Cubic spline interpolation of cross section vs neutron energy ')
legend ('data','spline')
```

Result

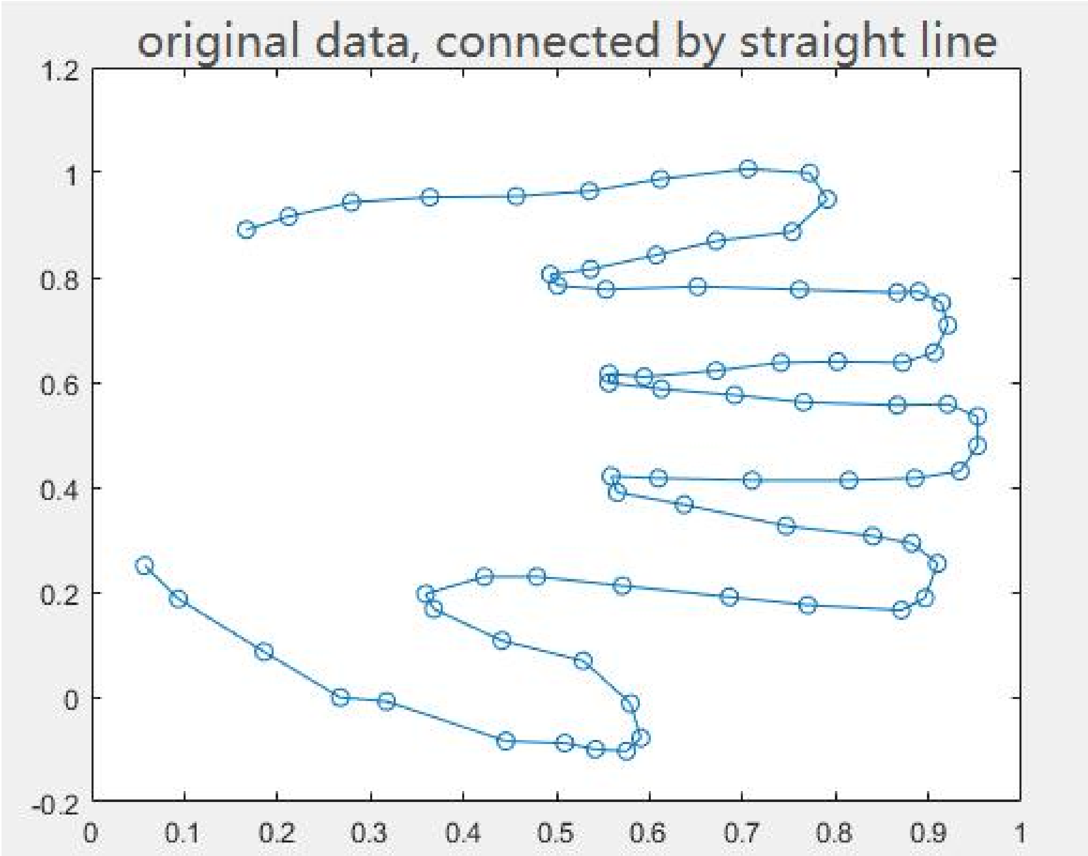
Result

The predicted cross section are \begin{array}{c|lcr} E & f(E) \ \hline 10& 10.8384 \ 90& 70.4942 \ 185& 7.1440 \end{array} The plot is as followings:

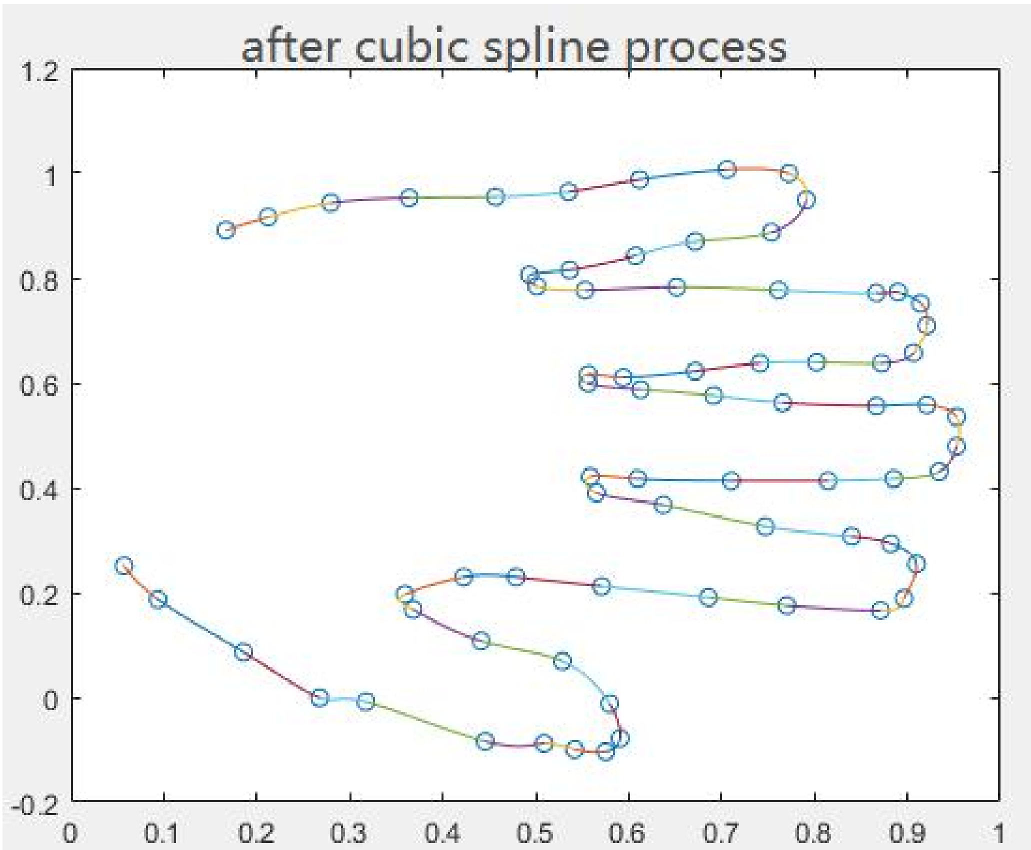


2

We firstly use ginput to get a set of data of Leon's left hand contour.(73 points totally)



Then we use cubic spline to interpolate (n against x, n against y)(n=1,...73), and plot x against y, we get



It's much smoother.

```
%cubic spline for X, Y row vector (n+1 dimension) (X in increasing
order)
%input (X, Y) for Natural Boundary Condition ( ''=0 at two ends)
%input (X, Y, p, q) for Clamped Condition ( left'=p, right'=q)
%return n set of parameters ( d(i),c(i),b(i),a(i) )
%where yi=di+ci(x-xi)+bi(x-xi)^2+ai(x-xi)^3
%Periodic Condition may be added to it later
%plot of cubicsplinized data is available (%in code)
narginchk(2,4);

%Natural Boundary Condition
if nargin == 2
    X=varargin{1};
    Y=varargin{2};
    [~,n]=size(X);
    n=n-1;

    H=X(2:n+1)-X(1:n);
    D=Y;
    B=zeros(1,n+1);
    A=zeros(1,n);
    C=A;

    M=zeros(n+1,n+1);
    N=zeros(1,n+1);

    M(1,1)=1; N(1)=0;
    M(n+1,n+1)=1; N(n+1)=0;

    for i=2:n
        M(i,i-1)=H(i-1);
        M(i,i)=2*(H(i)+H(i-1));
        M(i,i+1)=H(i);
        N(i)=3*(D(i+1)-D(i))/H(i)-3*(D(i)-D(i-1))/H(i-1);
    end

    B=thomas_tridiagonal(M,N);
    A=(B(2:n+1)-B(1:n))/3./H;
    C=(D(2:n+1)-D(1:n))./H-H.*(2*B(1:n)+B(2:n+1))/3;

%     plot(X,Y,'o');
%     hold on
%     for i=1:n
%         x=linspace(X(i),X(i+1),100);
%         y=D(i)+C(i)*(x-X(i))+B(i)*(x-X(i)).*(x-X(i))+A(i)*(x-X(i)).*
(x-X(i)).*(x-X(i));
%         plot(x,y);
%     end
```

```
if nargin == 4
    X=varargin{1};
    Y=varargin{2};
    p=varargin{3};
    q=varargin{4};
    [~,n]=size(X);
    n=n-1;

    H=X(2:n+1)-X(1:n);
    D=Y;
    B=zeros(1,n+1);
    A=zeros(1,n);
    C=A;

    M=zeros(n+1,n+1);
    N=zeros(1,n+1);

    M(1,1)=2*H(1);
    M(1,2)=H(1);
    N(1)=3*(D(2)-D(1))/H(1)-3*p;
    M(n+1,n)=H(n);
    M(n+1,n+1)=2*H(n);
    N(n+1)=3*q-3*(D(n+1)-D(n))/H(n);

    for i=2:n
        M(i,i-1)=H(i-1);
        M(i,i)=2*(H(i)+H(i-1));
        M(i,i+1)=H(i);
        N(i)=3*(D(i+1)-D(i))/H(i)-3*(D(i)-D(i-1))/H(i-1);
    end

    B=thomas_tridiagonal(M,N);
    A=(B(2:n+1)-B(1:n))/3./H;
    C=(D(2:n+1)-D(1:n))./H-H.*(2*B(1:n)+B(2:n+1))/3;

%     plot(X,Y,'o');
%     hold on
%     for i=1:n
%         x=linspace(X(i),X(i+1),100);
%         y=D(i)+C(i)*(x-X(i))+B(i)*(x-X(i)).*(x-X(i))+A(i)*(x-X(i)).*
(x-X(i)).*(x-X(i));
%         plot(x,y);
%     end
end
```

Code:Thomas method to solve tridiagonal matrix

```
%A is a tridigagonal matrix (n,n), D is a row vecter (1,n)
%AX=D
[~,n]=size(D);
X=zeros(1,n);
A(1,2)=A(1,2)/A(1,1);
for i=2:n-1
    A(i,i+1)=A(i,i+1)/(A(i,i)-A(i,i-1)*A(i-1,i));
end
D(1)=D(1)/A(1,1);
for i=2:n
    D(i)=(D(i)-A(i,i-1)*D(i-1))/(A(i,i)-A(i,i-1)*A(i-1,i));
    end
X(n)=D(n);
for i=n-1:-1:1
    X(i)=D(i)-A(i,i+1)*X(i+1);
end
```

Code:plot hand

```
%X,Y is data points of the hand contour(73 points totally)
    plot(X,Y,'o');
    hold on;
    nn=1:73;
[Dx,Cx,Bx,Ax]=cubicspline(nn,X);
[Dy,Cy,By,Ay]=cubicspline(nn,Y);
    [~,n]=size(X);
    n=n-1;
    for i=1:n
        x=linspace(nn(i),nn(i+1),100);
        fx=Dx(i)+Cx(i)*(x-nn(i))+Bx(i)*(x-nn(i)).*(x-nn(i))+Ax(i)*(x-
nn(i)).*(x-nn(i)).*(x-nn(i));
        fy=Dy(i)+Cy(i)*(x-nn(i))+By(i)*(x-nn(i)).*(x-nn(i))+Ay(i)*(x-
nn(i)).*(x-nn(i)).*(x-nn(i));
        plot(fx,fy);
    end
```

3

3

（a）

According to the formula, the round-off error of $y(t \pm h)$ can be calculated in this way:

$\frac{d\cos(t)}{dt} = -\sin(t)$ $E_{round}(h) = \frac{e_1 - e_{-1}}{2h}$ $E_{approx}(h) = \frac{h^2 f^{(3)}(\xi)}{6} \leq \frac{Mh^2}{6}$ where $M = max(|f^{(3)}(t)|) = \sin(t)$

So we can calculate e_1, e_{-1}, E_{round}, E_{approx}, E respectively.

h	e1 (1.0e-15)	e -1 (1.0e-16)	Eround	Eapprox	E
0.1	-0.10883603	-0.78268458	9.35522417e-16	0.0014024516	0.0014024516
0.01	-0.094955424	-0.91887448	9.34214361e-15	1.40245164e-05	1.40245164e-05
0.001	-0.093573526	-0.93266718	9.34201220e-14	1.40245164e-07	1.40245258e-07
0.0001	-0.093435449	-0.93404768	9.34201089e-13	1.40245164e-09	1.40338584e-09

From the table, we can see $\epsilon = \frac{1}{2}(|e_1| + |e_{-1}|) \approx 1e-16$. So the value of h that minizies the total error is $h = (\frac{3\epsilon}{M})^{1/3} \approx 7e-6$

What's more, we calculate the real error of central-difference algorithm $E_{real} = \frac{y_1 - y_{-1}}{2h} - y'(t) = \frac{cos(1+h) - cos(1-h)}{2h} + sin(1)$ for h = [1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6]

```
E_real =
0.001401750582461
0.000014024446294
0.000000140245188
0.000000001402529
0.000000000010864
0.000000000027517
```

We can see that the minial error lays around 1e-5, which accords to our estimation.

Code:

```
%第五次作业
%第三题(1)

clear all
clc
format long
t = 1;
epsm = 1.1102e-16; %机器误差
h = [0.1,0.01,0.001,0.0001,0.00001,0.000001]; %步长
e_plus = -sin(t+h).*(t+h).*epsm; %e1
e_minus = -sin(t-h).*(t-h).*epsm; %e-1
epsmax = (-e_plus - e_minus)./2 ;

E_round = (-e_plus - e_minus)./(2.*h); %E_round
M = abs(sin(t));
E_approx = M*h.^2/6; %E_approx
E = E_approx + E_round; %通过公式估算出来的误差
h_min = (3*epsmax./M).^(1/3); %通过公式估算出来使误差最小的h
```

(b)

For $\epsilon = 0.5 \times 10^{-8}$, the optimal value of step size $h = (\frac{3\epsilon}{M})^{1/3} = (\frac{3\times0.5\times10^{-8}}{\sin(1)})^{1/3} = 2.612 \times 10^{-3}$ Which is lager than Problem 1, because the magnitude of ϵ we take in Problem 1 is 1e-16 and we have already discussed that the optimal value of step size h is about 7e-6

(c)

Using Taylor expansion:
$y(t+h) = y(t) + hy^{(1)}(t) + \frac{h^2}{2}y^{(2)}(t) + \frac{h^3}{6}y^{(3)}(t) + \frac{h^4}{24}y^{(4)}(t) + \frac{h^5}{120}y^{(5)}(\xi)$

$y(t-h) = y(t) - hy^{(1)}(t) + \frac{h^2}{2}y^{(2)}(t) - \frac{h^3}{6}y^{(3)}(t) + \frac{h^4}{24}y^{(4)}(t) - \frac{h^5}{120}y^{(5)}(\xi)$

$y(t+2h) = y(t) + 2hy^{(1)}(t) + 2h^2y^{(2)}(t) + \frac{4h^3}{3}y^{(3)}(t) + \frac{2h^4}{3}y^{(4)}(t) + \frac{4h^5}{15}y^{(5)}(\xi)$

$y(t-2h) = y(t) - 2hy^{(1)}(t) + 2h^2y^{(2)}(t) - \frac{4h^3}{3}y^{(3)}(t) + \frac{2h^4}{3}y^{(4)}(t) - \frac{4h^5}{15}y^{(5)}(\xi)$

then $\frac{-y(t+2h)+8y(t+h)-8y(t-h)+y(t-2h)}{12h} = \frac{12hy^{(1)}(t)-2/5h^5y^{(5)}(\xi)}{12h} = y^{(1)}(t) - \frac{1}{30}h^4y^{(5)}(\xi)$

So $\mathbf{E_{approx}} = \frac{\mathbf{h^4}}{\mathbf{30}}\mathbf{y^{(5)}}(\xi)$

And $y(t+h) = y_1 + e_1$ $y(t-h) = y_1 + e_{-1}$ $y(t+2h) = y_1 + e_2$ $y(t-2h) = y_1 + e_{-2}$

So $\mathbf{E_{round}} = \frac{\mathbf{-e_2+8e_1-8e_{-1}+e_{-2}}}{\mathbf{12h}} \leq \frac{\mathbf{3}\epsilon}{\mathbf{2h}}$

$|\mathbf{E(h)}| \leq \frac{\mathbf{3}\epsilon}{\mathbf{2h}} + \frac{\mathbf{1}}{\mathbf{30}}\mathbf{h^4y^{(5)}}(\xi)$

the value of h that minimizes the total error is $h = (\frac{45\epsilon}{4M})^{1/5}$ where $M = max(|f^{(5)}(x)|)$

4

(a)

Three-point backward difference formula: $f'(x_i) = \frac{f(x_{i-2})-4f(x_{i-1})+3f(x_i)}{2h} + O(h^2)$ So $f'(2010) = \frac{f(1990)-4f(2000)+3f(2010)}{2*10} = 0.345$

(b)

Two-point central difference formula: $f'(x_i) = \frac{f(x_{i+1})-f(x_{i-1})}{2h} + O(h^2)$ So we can rewrite $f'(2010) = \frac{f(2020)-f(2000)}{2*10}$ So we get $f(2020) = 20 * 0.345 + 30.8 = 37.7$