The First 2 functions are for grading, the rest two are written to verify the results.

```
%polynimoial_piecewise_error_with_derivatives
%Using Derivatives to calculate the error.
function [h,inferror,i]=Polynimial_piecewise_error_with_derivatives(n)
clf
format long
h=10/n;
x=[-5:10/n:5];
y=1./(1+x.^2);
x0=[-5:0.005:5];
y0=1./(1+x0.^2);
y1=1./(1+x.^2);

plot(x,y,'--r')
%Interpolation
hold on
%Origin
plot(x0,y0,'-b')
    for k=1: n
            slopei=(y(k)-y(k+1))/(x(k)-x(k+1));
            ki=y(k)-slopei*x(k);
            F = @(x)  slopei*(1+x^2)*(1+x^2)+2*x;
            maxpoint=fzero(F,x(k));
            yi=slopei*maxpoint+ki;
            y0=1./(1+maxpoint.^2);
            error(k)=abs(yi-y0);
    end
h=10/n
[inferror,i] = max(error)
%i is the interval with get max error, inferror is the value of max error.
legend('Interpolation','Origin')
xlabel(['The error is',num2str(inferror)])
title(['Piecewise Linear Interpolation with
h=',num2str(h),'(',num2str(n),'points)'])
```

```
function [h0,inferror,i]=spline3(number)
%number is the interval you hope to get.
%h0 is the mesh size, inferror is the biggest error of the function,
%i is the number of interval achives the bigggest error.
clf
h0=10/number;
x=[-5:h0:5];
y=1./(1+x.^2);
x0=[-5:0.005:5];
y0=1./(1+x0.^2);
%Origin
plot(x0,y0,'g')
hold on
n=length(x);% n=number+1
h=diff(x);
h(n)=h(n-1);
d = zeros(n,1);
 %Rows 2 to (n-1) of d, (bn on the PPT)
 for i = 2:n-1
    d(i) = (1/h(i))*y(i-1) - ((1/h(i)) + (1/h(i+1)))*y(i) +
(1/h(i+1))*y(i+1);
 end
 %Rows 1 and n of b
 d(1) = (2*x(1))/((1+x(1)^2)^2) - (1/h(1))*y(1) + (1/h(1))*y(2);
 d(n) = -(2*x(n))/((1+x(n)^2)^2) + (1/h(n))*y(n-1) - (1/h(n))*y(n);
 d = 6*d;

A = zeros(n,n);
 %Rows 2 to (n-1) of A
 for i = 2:(n-1)
     A(i,i-1) = h(i-1); %Subdiagonal areas
     A(i,i) = 2*(h(i-1)+h(i)); %Diagonal areas
     A(i,i+1) = h(i); %Upper-diagonal areas
 end
 %Rows 1 and n of A
 A(1,1) = 2*h(1);
 A(1,2) = h(1);
 A(n,n) = 2*h(n-1);
 A(n,n-1) = h(n-1);
M=inv(A)*d;   %Solve the Matrix
syms g
```

```
for k=1:n-1   %Solve for the spline
s(k)=M(k)*(x(k+1)-g)^3/(6*h(k))+M(k+1)*((g-x(k))^3/(6*h(k)))+(y(k)-M(k)
*h(k)^2/6)*(x(k+1)-g)/h(k)+(y(k+1)-M(k+1)*h(k)^2/6)*(g-x(k))/h(k)+g^3;
end
for k=1:n-1
    S(k,:)=sym2poly(s(k));  %piecewise splines
end
%fix a calculation problem when n=5
E = zeros(n-1,4);
E(:,1)=1;
a=S-E;
for i=1:2000 %Calculate the error approximately.
    for k=1: n-1
        if x0(i)>=x(k)&x0(i)<x(k+1)
            yi(i)=a(k,1).*x0(i).^3 + a(k,2).*x0(i).^2 + a(k,3).*x0(i)+
a(k,4);
            truey=1./(1.+x0(i).^2);
            error(i)=abs(yi(i)-truey);
        end
    end
end
yi(2001)=a(n-1,1).*x0(2001).^3 + a(n-1,2).*x0(2001).^2 +
a(n-1,3).*x0(2001)+ a(n-1,4);
error(2001)=abs(yi(2001)-y0(2001)); %Do not forget the last elements!.
plot(x0,yi,'r')
inferror=max(error);
xlabel(['The error is',num2str(inferror)]) %output everything in figure.
legend('Origin','Interpolation')
title(['Piecewise Linear Interpolation with
h=',num2str(h0),'(',num2str(n),'points)'])
```

```
%polynimoial_piecewise
%Using 2000 points to estimate the error
function [h,inferror]=polynimoial_piecewise(n)
clf
h=10/n;
x=[-5:10/n:5];
y=1./(1+x.^2);
x0=[-5:0.005:5];
y0=1./(1+x0.^2);
y1=1./(1+x.^2);
plot(x,y,'--r')
%Interpolation
hold on
%Origin
plot(x0,y0,'-b')
for i=1:2000
    for k=1: n
        if x0(i)>=x(k)&x0(i)<x(k+1)
            slopei=(y(k)-y(k+1))/(x(k)-x(k+1))
            ki=y(k)-slopei*x(k)
            yi=slopei*x0(i)+ki
            error(i)=abs(yi-y0(i))
        end
    end
end
inferror=max(error)
legend('Interpolation','Origin')
xlabel(['The error is',num2str(inferror)])
title(['Piecewise Linear Interpolation with
h=',num2str(h),'(',num2str(n),'points)'])
```

```matlab
%Clamped_Cubic_Spline
%Using 2000 points to approximate the error with plug_in function spline().
function [h]=Clamped_Cubic_Spline(n)
clf
h=10/n
x = -5.:h:5.;
y = 1./(1+x.^2);
xx = -5.:h/500:5.;
yy = spline(x,y,xx);
plot(xx,yy,'r')
hold on
x0=[-5:0.005:5];
y0=1./(1+x0.^2);
plot(x0,y0,'-b')
legend('Interpolation','Origin')
x1 = x;
y1=1./(1+x1.^2)
pp=spline(x1,y1)
a=pp.coefs
for i=1:2000
          yi=spline(x1,y1,x0(i))
          error(i)=abs(yi-y0(i))
end
inferror=max(error)
xlabel(['The error is',num2str(inferror)])
title(['Cubic Spline Interpolation with
h=',num2str(h),'(',num2str(n),'points)'])
```