MATH 226, HOMEWORK 1

SHENG XU

(1) Bernstein polynomials

(a) Proof: Linearity of Bernstein polynomials

$$\begin{aligned} B_n(\alpha f + \beta g) &= \sum_{k=0}^{n} (\alpha f(\frac{k}{n}) + \beta g(\frac{k}{n})) \binom{n}{k} x^k (1-x)^{n-k} \\ &= \sum_{k=0}^{n} \alpha f(\frac{k}{n}) \binom{n}{k} x^k (1-x)^{n-k} + \sum_{k=0}^{n} \alpha g(\frac{k}{n}) \binom{n}{k} x^k (1-x)^{n-k} \\ &= \alpha B_n f + \beta B_n g. \end{aligned}$$

(b) Positive Operator: We first prove if $f \geq g$, then $B_n f \geq B_n g$. Then set $g \equiv 0$, we get $B_n g = 0$. And we can prove that if $f \geq 0$ then $B_n f \geq 0$.

$$B_n f - B_n g = \sum_{k=0}^{n} (f(\frac{k}{n}) - g(\frac{k}{n})) \binom{n}{k} x^k (1-x)^{n-k}$$

Since $f \geq g$, $f - g \geq 0$. With $f - g \geq 0$, $\binom{n}{k} \geq 0, x \geq 0, (1-x) \geq 0$, We have $B_n f - B_n g \geq 0$.
Thus, we proved that if $f \geq g$, then $B_n f \geq B_n g$.
Let's set $g \equiv 0$. We get $B_n g = 0$. In that case, if $f \geq 0, B_n f \geq 0$. So B_n is positive operator.

(c) Let $f_0 = 1$, $f_1 = x$, and $f_2 = x^2$, show that $B_n f_0 = f_0$, $B_n f_1 = f_1$, and $B_n f_2 = \frac{n-1}{n} f_2 + \frac{1}{n} f_1$.

$$\mathcal{P}_n(I) = Span(1, x, x^2), f_0 = 1, f_1 = x, f_2 = x^2$$

$$B_n f_0 = \sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k} = [x + (1-x)]^n = 1 = f_0$$

$$\begin{aligned} B_n f_1 &= \sum_{k=0}^{n} \frac{k}{n} \binom{n}{k} x^k (1-x)^{n-k} \\ &= \sum_{k=1}^{n} \binom{n-1}{k-1} x^{k-1} (1-x)^{n-k} * x \\ &= [x + (1-x)]^{n-1} * x = x = f_1 \end{aligned}$$

$$\begin{aligned} B_n f_2 &= \sum_{k=0}^{n} \frac{k^2}{n^2} \binom{n}{k} x^k (1-x)^{n-k} \\ &= \sum_{k=0}^{n} \frac{k^2 - k + k}{n^2} \binom{n}{k} x^k (1-x)^{n-k} \\ &= \sum_{k=0}^{n} \frac{k^2 - k}{n^2} \binom{n}{k} x^k (1-x)^{n-k} + \sum_{k=0}^{n} \frac{k}{n^2} \binom{n}{k} x^k (1-x)^{n-k} \\ &= \frac{n^2 - n}{n^2} x^2 \sum_{k=2}^{n} \binom{n-2}{k-2} x^{k-2} (1-x)^{n-k} + \frac{1}{n} x \sum_{k=1}^{n} \binom{n-1}{k-1} x^{k-1} (1-x)^{n-k} \\ &= \frac{n-1}{n} x^2 * [x + (1-x)]^{n-2} + \frac{1}{n} x * [x + (1-x)]^{n-1} \\ &= \frac{n-1}{n} x^2 + \frac{1}{n} x \\ &= \frac{n-1}{n} f_2 + \frac{1}{n} f_1 \end{aligned}$$

(2) Proof: the Chebyshev Alternation Theorem

(a) We first prove that "if $f - p^*$ achieves its maximun magnitude at $n+2$ distinct point with alternating sign, then p^* is the best approximation."
We have proved Kolmogorov Characterization Theorem that Let $f \epsilon C(I)$ and P be a finite dimensional subspace of C(I), then p^* in P is a best approximation to f if and only if no element of P has the same sign as $f - p^*$ on its extreme set.
So we can prove if no element of P has the same sign as $f - p^*$ on its extreme set, then $f - p^*$ achieves its maximun magnitude at $n+2$ distinct point with alternating sign. Here we can use contradiction to prove this easily.
$\forall q \epsilon \mathcal{P}_n, \therefore q = 0$ has at most n roots.
If $[f - p^*]q > 0$ and $f - p^*$ achieves its maximun magnitude at $n+2$ distinct point with alternating sign, q has to change its sign on alternative sets for $n+2$ times. According to the intermediate Value Theorem, q=0 will have

n+1 roots, which is a contradiction.
Therefore, for any $q\epsilon\mathcal{P}_n$, $f - p^*$ does not have the same sign with q, which satisfies the Kolmogorov Characterization Theorem. Thus, p^* is the best approximation.

(b) Now, We prove that "if p* is the best approximation, then $f - p^*$ achieves its maximun magnitude at $n + 2$ distinct point with alternating sign."
If $f - p^*$ achieves its maximun magnitude at less than $n + 2$ distinct point with alternating sign So we set $m \leq n + 1$
We construct 2 sets from the extreme set:

$$\mathcal{Z}_{\pm} = \{x\epsilon[0,1]|f(x) - p*(x) = \pm\|f - p\|\}$$

Therefore, there exist an ordered set of m disjoint interval$\{K_i\}, K_i = (k_{i-1}, k_i), k_i\epsilon R$, which contains both $\mathcal{Z}_{\pm}$ and such that on adjacant intervals, the points from the extreme set belongs to $\mathcal{Z}_+$ and $\mathcal{Z}_-$ alternatively. (That means in every disjoint interval K_i, you can have one or several x_i such that $[f(x_i) - p^*(x_i)]$ maintains the same sign while on the next interval K_{i+1} you must have one or several elements x_j bigger than $\max(x_i)$ in the K_i and $[f(x_j) - p^*(x_j)]$ do not have the same sign with those in K_i).Select one point z_k ($k = 1, 2, \ldots, m$ since we have m alternating points.)from every adjacent sets K_i
In this case, we have $k_0 < z_1 < k_1 < z_2 < \ldots < k_{m-1} < z_m$, with which we can construct

$$q = \Pi_{k=0}^{n}(x - k_i), i = 1, 2, \ldots, m - 1.$$

since $m - 1 \leq n$, $q\epsilon\mathcal{P}_n$.
For this q (or -q), we'll have $[f - p^*]q > 0$ (or $[f - p^*](-q) > 0$) at extreme set, which reject Kolmogorov Characterization Theorem. Thus we prove p* is the best approximation leads to $f - p^*$ achieves its maximun magnitude at $n + 2$ distinct point with alternating sign.

(3) Error Formula for the Hermite interpolation
Proof: To simplify the function, let's define:

$$W(x) = \Pi_{k=0}^{n}(x - x_i)^2$$

$\forall xin[a, b]$ and $x \neq x_i$, Here, we define an function

$$g(t) = f(t) - \mathcal{P}_{2n+1}(t) - \frac{f(x) - \mathcal{P}_{2n+1}(x)}{W(x)}W(t)$$

If we take $2n+2$ derivatives of g(t), we have:

$$g^{(2n+2)}(t) = f^{(2n+2)}(t) - 0 - \frac{f(x) - \mathcal{P}_{2n+1}(x)}{W(x)}(2n+2)!$$

So we need to find an ξ such that $g^{(2n+2)}(\xi) = 0$ to finish the proof. With the definition of Hermit interpolation:

$$f(x_i) = \mathcal{P}_{2n+1}(x_i), i = 0, 1, 2, 3, \ldots, n$$
$$f'(x_i) = \mathcal{P}'_{2n+1}(x_i), i = 0, 1, 2, 3, \ldots, n$$

Note that we have $n+2$ roots $(x, x_0, x_1, x_2, \ldots, x_n)$ for $g(t) = f(t) - \mathcal{P}_{2n+1}(t) - \frac{f(x) - \mathcal{P}_{2n+1}(x)}{W(x)}W(t)$ So with Rollers Rule, we can show there exist $\ell_k (k = 0, 1, \ldots, n)$ and $\ell_k \neq x_i$ (k=0,1,. . . ,n; i=0,1,. . . ,n) such that $g'(\ell_k) = 0$
With $g'(x_i) = 0, i = 0, 1, 2, 3, \ldots, n$, we find $2n+2$ roots $(x_i, i = 0, 1, 2, 3, \ldots, n$ and $\ell_k, k = 0, 1, \ldots, n)$ for g'(t)=0. With Rollers Rule, we can find an ξ, makes $g^{(2n+2)}(\xi)$=0. Since $\ell_k (k = 0, 1, \ldots, n)$ are related with $x_i (i = 0, 1, 2, 3, \ldots, n)$, So ξ is related with $x_i (i = 0, 1, 2, 3, \ldots, n)$.
Last, we take $g^{(2n+2)}(\xi)$=0 into the equation and solve for $f(x) - \mathcal{P}_{2n+1}(x)$

$$0 = g^{(2n+2)}(t) = f^{(2n+2)}(t) - 0 - \frac{f(x) - \mathcal{P}_{2n+1}(x)}{W(x)}(2n+2)!$$

We can get the conclusion.

$$f(x) - p_{2n+1}(x) = \frac{1}{(2n+2)!} f^{(2n+2)}(\xi) \left[(x - x_0)(x - x_1) \cdots (x - x_n)\right]^2,$$

(4) Error Formula for the Hermite interpolation
Let $I = [-1, 1]$ and $\mathcal{P}_n(I)$ be the space of polynomial functions of degree at most n on I. For any $q \in \mathcal{P}_n(I)$, show that $\|q\|_\infty \leq K(n)\|q\|_2$ with $K(n) = \frac{n+1}{\sqrt{2}}$.
Proof: we define $\{\phi_n\}$ as the Lagendre polynomials, which is an orthogonal basis of the Polynomial space L^2 with the property:
$\|\phi_n\|_{C(I)} = 1$
$(\phi_m, \phi_n)_{L^2(I)} = \frac{1}{(2n+1)}\delta_{mn}$

For $q \epsilon P_n$, we can expand q with Lagendre polynomials, let $q = \sum_{i=1}^{n} \alpha_i \cdot \phi_i$. So we have $\|q\|_{L^2(I)} = \sum_{i=1}^{n} \frac{1}{(2n+1)}\|\alpha_i\|^2$

$$
\begin{aligned}
\|q\|_\infty &= \|\sum_{i=1}^{n} a_i\phi_i\| \le \sum_{i=1}^{n} \|a_i\|\|\phi_i\| \\
&\le \sum_{i=1}^{n} \|a_i\| \ (Because|\phi_i\| < 1) \\
&= \sum_{i=1}^{n} \|a_i\| \sqrt{\frac{2}{2i+1}} \sqrt{\frac{2i+1}{2}} \\
&= \sqrt{\sum_{i=1}^{n} a_i^2 \frac{2}{2i+1}} \sqrt{\sum_{i=1}^{n} \frac{2i+1}{2}} (ByCauchy - Schwarz) \\
&= \|q\|_2^2 \sqrt{\frac{n(n+2)}{2}} = \|g\|_2^2 \sqrt{\frac{n^2+2n}{2}} \\
&\le \|g\|_2^2 \sqrt{\frac{(n^2+2n+1)}{2}} \\
&= \|q\|_2^2 K(n)
\end{aligned}
$$

(5) Consider the function $f(x) = \dfrac{1}{1+x^2}$. Write codes to find the piecewise linear polynomial interpolation and clamped cubic spline to approximate $f(x)$ on $[-5, 5]$ with equally distributed points of mesh size h. Plot the approximations and observe how the errors change when the mesh size h decreases.
Result:From both table we can figure out that the error increases with h. The smaller h is, the more accurate the interpolation will be. The interpolation of Cubic Spline seem to be much more accurate than the piecewise linear polynomial interpolation, though the piecewise linear polynomial interpolation is also quite accurate when number of points grows much bigger.

Table 3-1 Relationship between mesh size h and Error of infinity norm of Piecewise Linear Approximation

h	2	1	0.5	0.2	0.1	0.05
$error$	0.50	0.067	0.042	0.0093	0.0025	0.00062

Table 3-2 Relationship between mesh size h and Error of infinity norm of Cubic Spline

h	2	1	0.5	0.2	0.1	0.05
$error1$	0.43	0.022	0.0032	0.00011	0.0000065	0.00000039
$error2$	0.42	0.022	0.0032	0.00011	0.0000065	0.00000039

Error1 is the error from plug-in function of Matlab, Error 2 is the function of mine

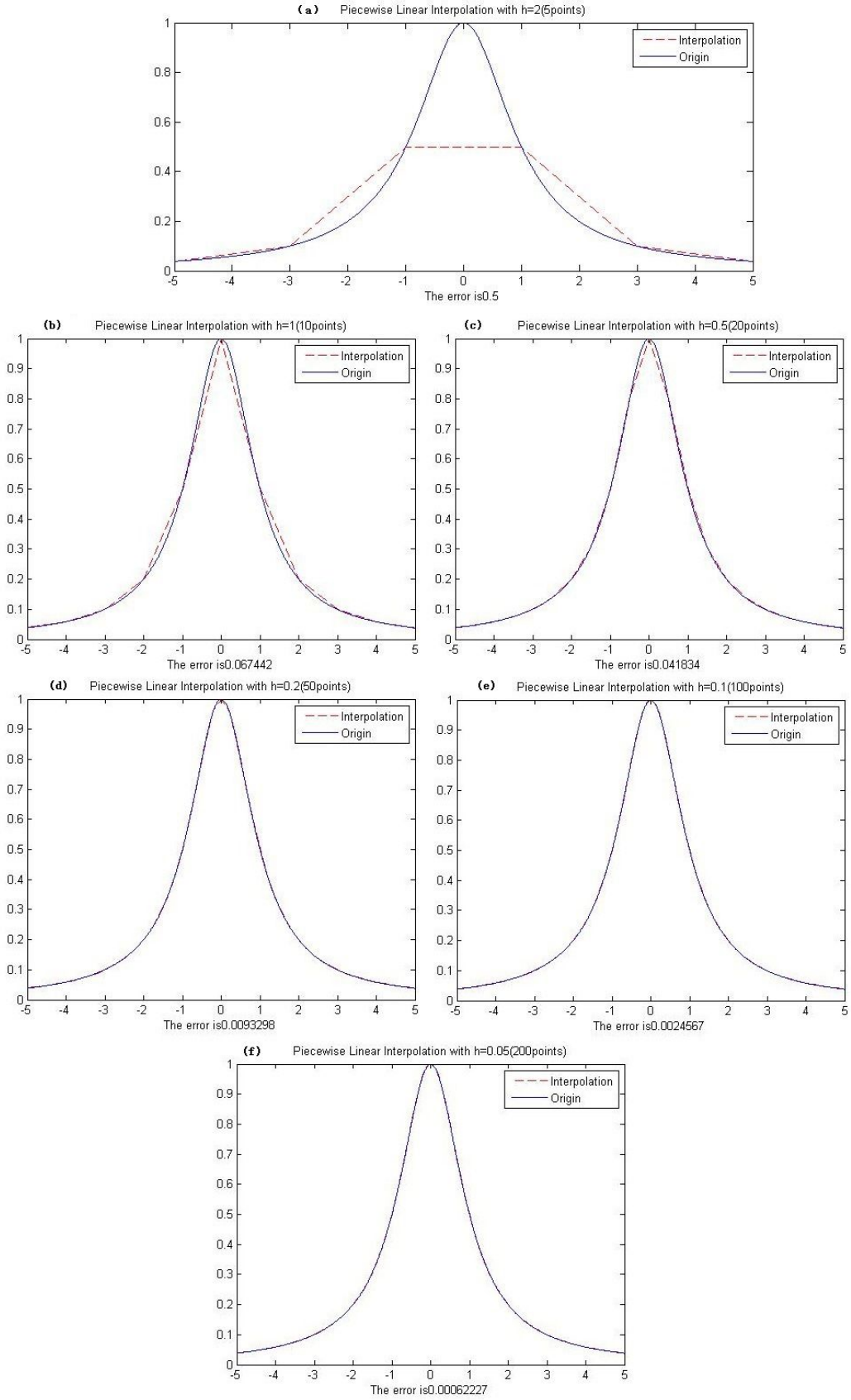

Figure 5.1 Piecewise Linear Interpolation with (a) h=2; (b) h=1; (c)h=0.5 (d) h=0.2; (e)h=0.1; (f) h=0.05

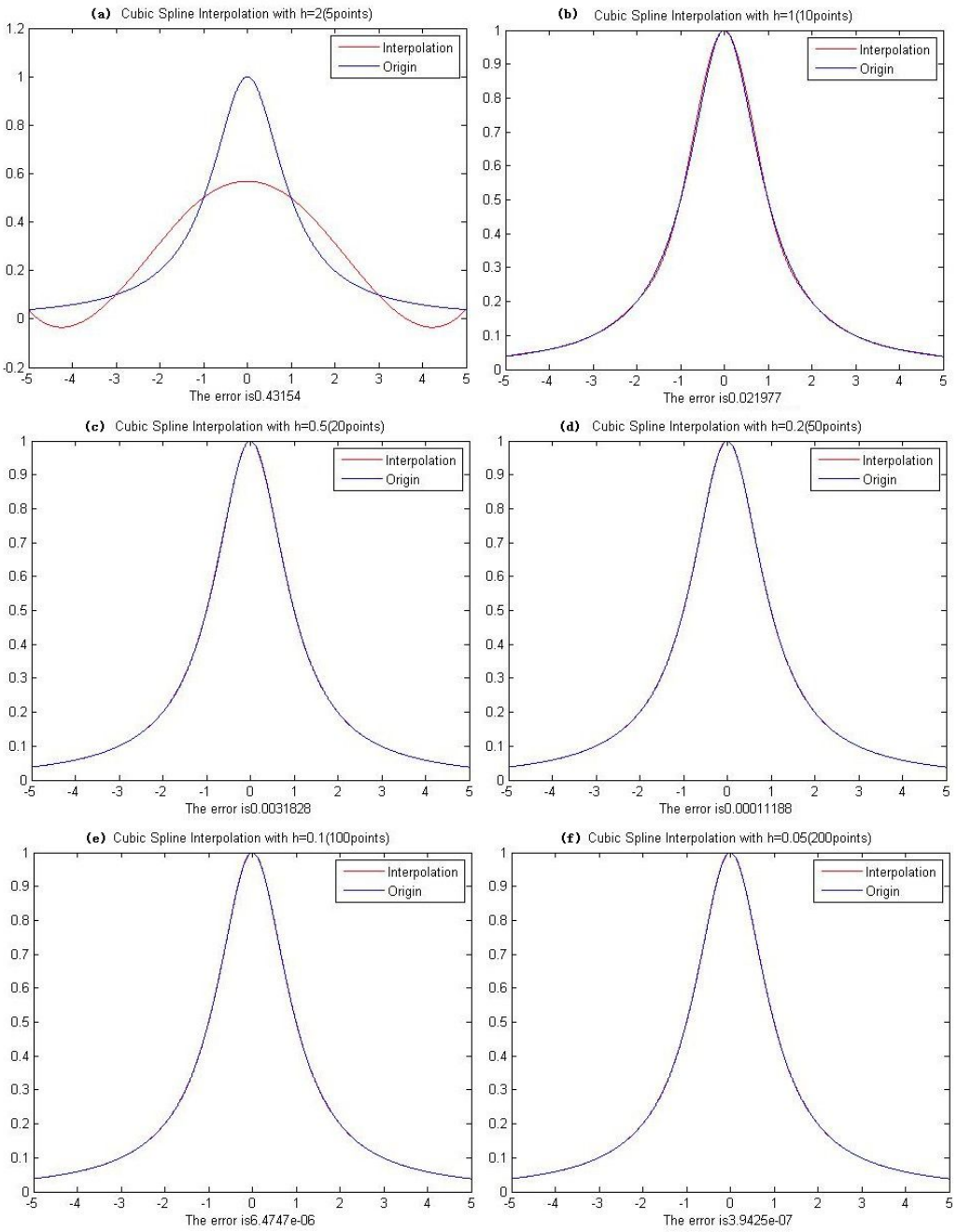

Figure 5.2 Piecewise Cubic Spline Interpolation with (a) h=2; (b) h=1; (c)h=0.5 (d) h=0.2; (e)h=0.1; (f) h=0.05

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)'])
```

```
%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)'])
```