Problem 4 section 3.2 computer problem 1

Function:

function c = newtdd(x,y,n)

for j = 1:n

v(j,1) = y(j);

end

for i = 2:n

for j = 1:n + 1 - i

v(j,i)=(v(j + 1,i - 1) - v(j,i - 1))/(x(j + i - 1) - x(j));

end

end

for i = 1:n

c(i) = v(1,i);

end

end

function p = value(a,x,c)

s = c(1);

n = length(c);

for i = 2:n

t = c(i);

for j = 1:i-1

t = t \* (a - x(j));

end

s = s + t;

end

p = s;

end

Main Code:

x = [0.6 0.7 0.8 0.9 1.0];

y = [1.433329 1.632316 1.896481 2.247908 2.718282];

n = 5;

c = newtdd(x,y,n);

a = 0.98;

p = value(a,x,c);

syms v

pv = c(1);

fv = exp(v^2);

for i = 2:n

t = c(i);

for j = 1:i-1

t = t \* (v - x(j));

end

pv = pv + t;

end

pv = vpa(expand(pv),5)

syms u

f = exp(u^2);

for i = 1:n

f = diff(f,u);

end

% u = linspace(0.6,1);

% plot(u,eval(f))

u = 1;

multi = 1;

for i = 1:n

multi = multi \* (a - x(i));

end

up\_bounds = abs(multi) \* abs(eval(f)) / factorial(n);

act\_error = abs(exp(a^2)-p);

v = linspace(0.5,1);

plot(v,eval(pv - fv))

v = linspace(1,2);

plot(v,abs(eval(pv - fv)))

1. The coeffcient of Newton’s divided differences formula are:

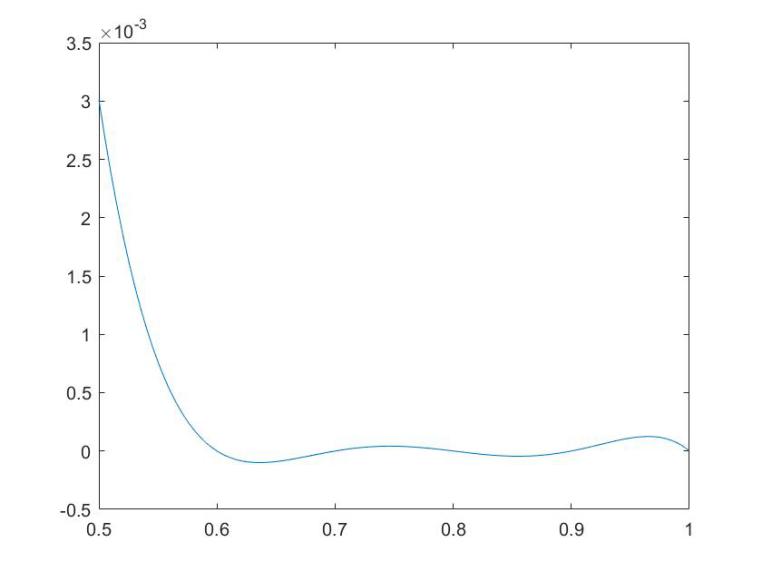
c = 1.433329000000000 1.989870000000001 3.258899999999984 3.680666666666721 4.000416666666682

And P4(x) = 4.0004\*x^4 - 8.3206\*x^3 + 8.9309\*x^2 - 3.4736\*x + 1.5812

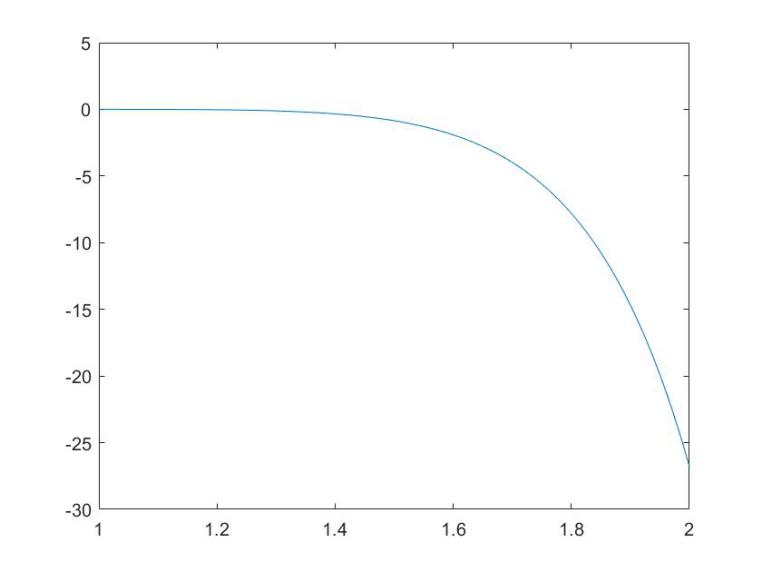
1. P4(0.82) = 1.958909774400000 and P4(0.98) = 2.612847966399999
2. When x = 0.82, the upper bound is 5.373586503516333e-05, and the actual error is 2.334851421492701e-05, we can see that actual error is smaller than the upper bound;

When x = 0.92, the upper bound is 2.165718196871742e-04, and the actual error is 1.066054239338143e-04, we can see that actual error is smaller than the upper bound;

1. For interval [0.5,1], the plot is as follows:



For interval [0,2], the plot is as follows:



Problem 7 section 3.3 computer problem 4

Function:

function y = nest(d,c,x,b)

if nargin<4, b = zeros(d,1); end

y = c(d+1);

for i = d:-1:1

y = y.\*(x - b(i)) + c(i);

end

end

Main Code:

n = 10;

leng = 2/n;

x1 = -1:leng:1;

y1 = exp(abs(x1));

c1 = newtdd(x1,y1,n+1);

a = -1:0.01:1;

p1 = nest(10,c1,a,x1);

plot(a,p1,'r')

hold on

x2 = cos((1:2:2\*(n+1)-1)\*pi/(2\*(n+1)));

y2 = exp(abs(x2));

c2 = newtdd(x2,y2,n+1);

p2 = nest(n,c2,a,x2);

p\_true = exp(abs(a));

error1 = norm(p1 - p\_true,'inf')

error2 = norm(p2 - p\_true, 'inf')

plot(a,p2,'b')

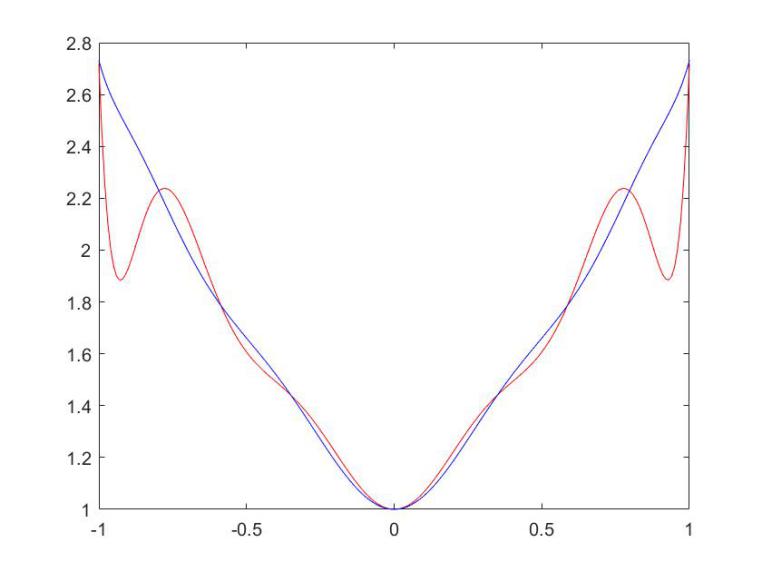
plot(a,p1 - p\_true,'r')

hold on

plot(a,p2 - p\_true,'b')

Answer:

When n is 10, the plot is as follows, the red one is evenly spaced interpolation and the blue one is Chebyshev interpolation



The backward error of two methods are:

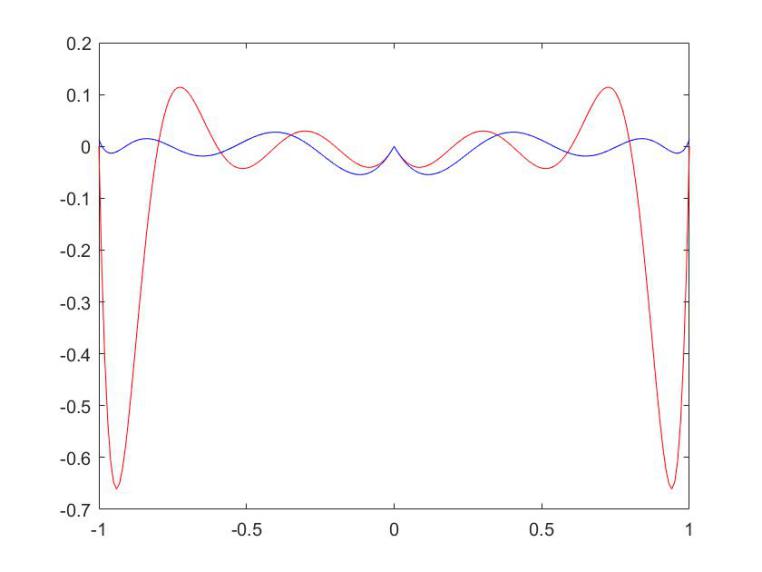
error1 =

0.660713755081157

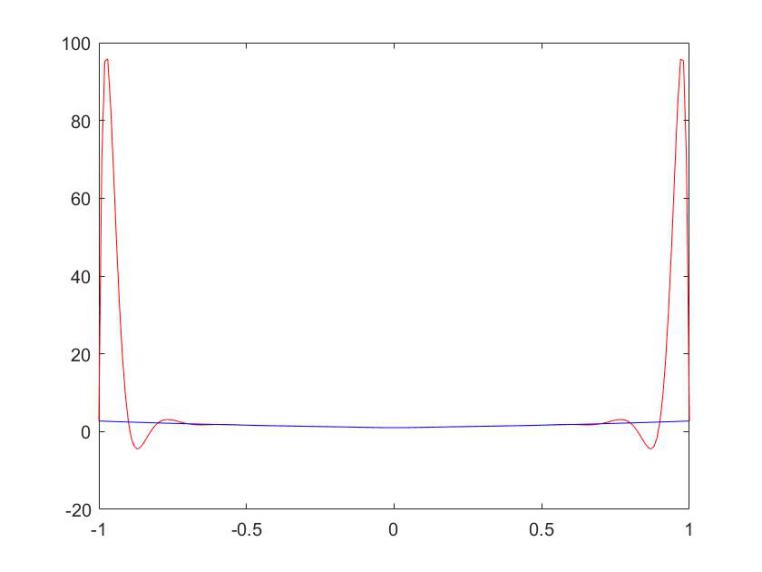
error2 =

0.054428276445230

We can see that evenly spaced interpolation has larger error than Chebyshev interpolation. The plot of the error interpolation is as follows:



And for n = 20, we can see the result as follows:



Again the red plot is evenly spaced interpolation and the blue one is Chebyshev interpolation.

The backward error:

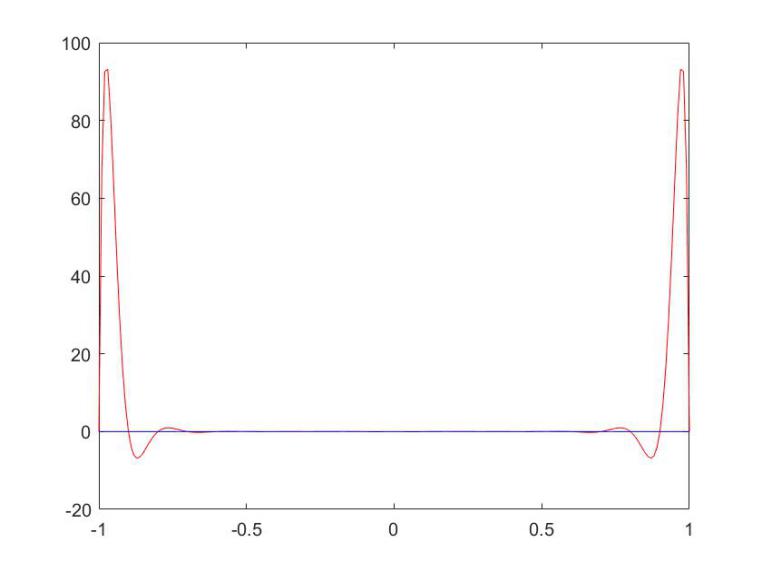
error1 =

93.164500012246734

error2 =

0.028458109952903

And the plot of error interpolation is as follows:

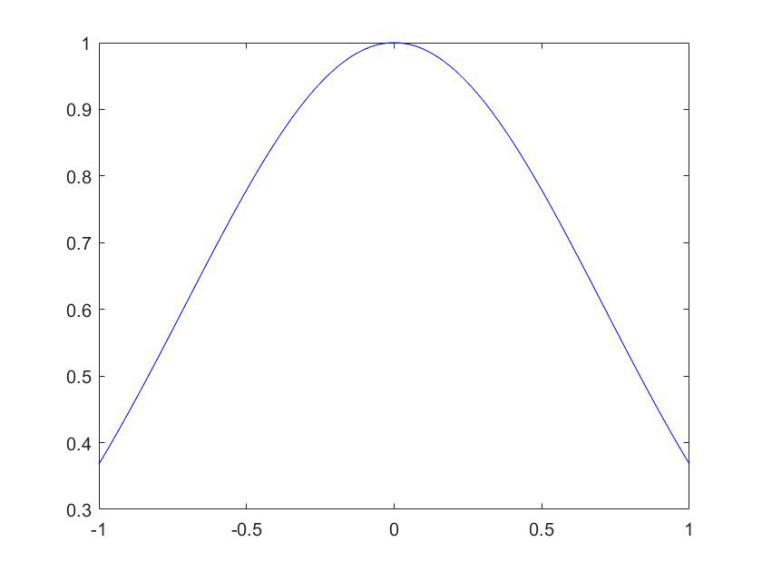


As the error of evenly spaced interpolation becomes larger quickly when n goes up, in conclusion, there is Runge phenomenon of evenly spaced interpolation.

section 3.3 computer problem 5

When n is 10

The plots of the two interpolation are as follows, we found that the two plot coincides with each other:



The backward error is as follows:

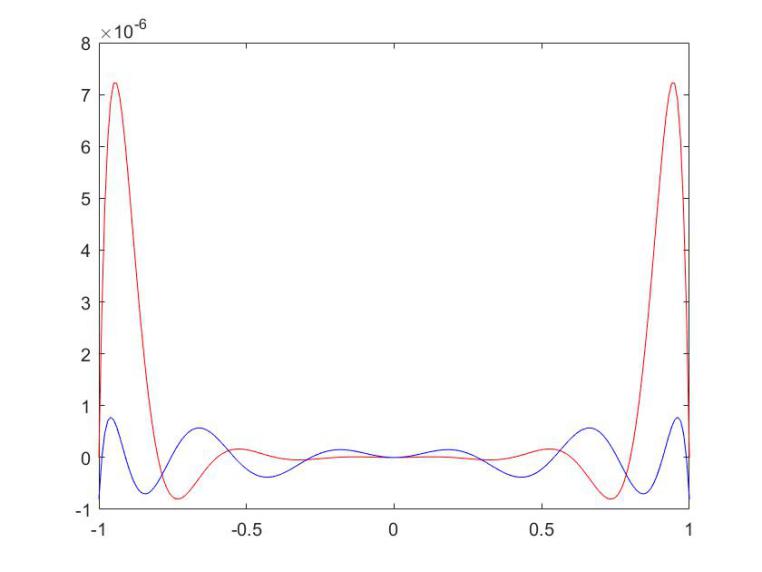
error1 =

7.222366145809289e-06

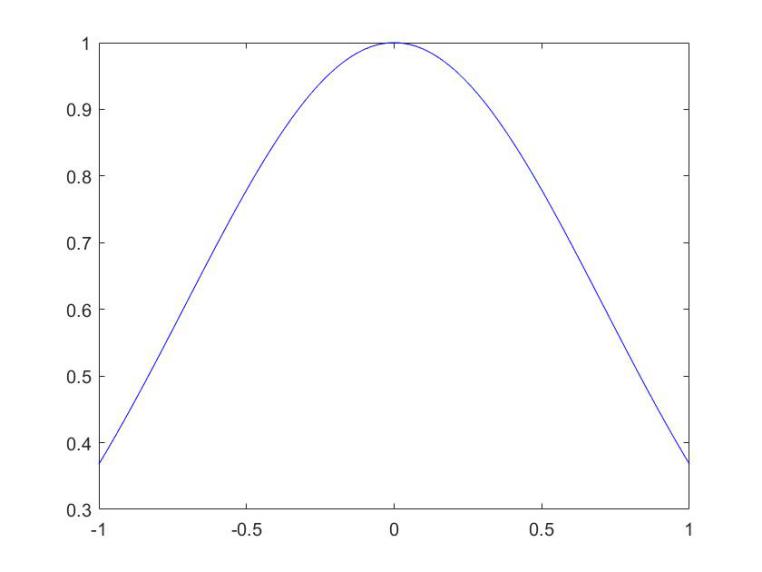
error2 =

8.013222552727406e-07

We can see that evenly spaced interpolation has larger backward error. The error interpolation is as follows:



When n is 20, the interpolation plots are as follows:



Still, the two plot concides with each other. Compute the backward error:

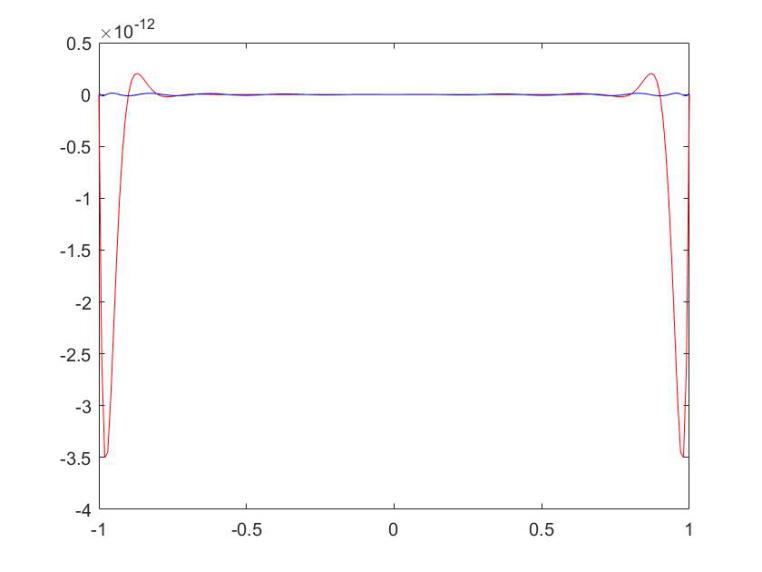
error1 =

3.501532397365281e-12

error2 =

1.454392162258955e-14

The evenly spaced interpolation error become smaller than before, but still larger than Chebyshev interpolation. The plots are as follows:



We found that with the increase of n, the error is smaller, so there is no Runge phenomenon.