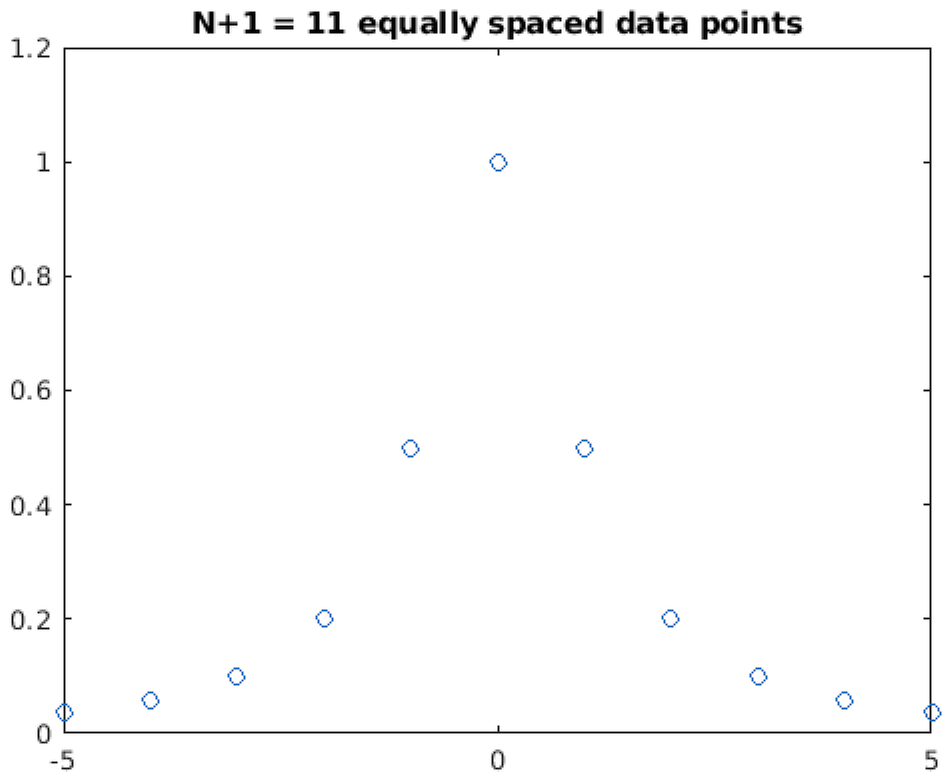
Table of Contents

MTH 351 LAB 3 John Waczak

```
clear all;
format long;
```

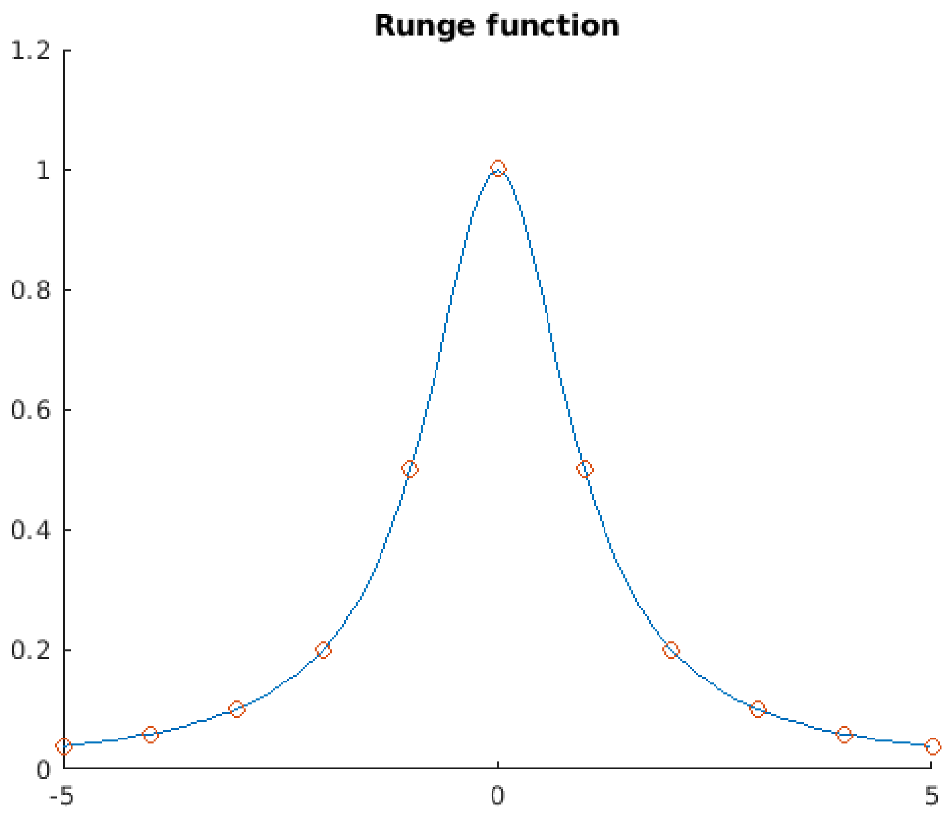
1.0 Problem Setup

```
        Generate N+1=11 equally speced nodes in [-5,5] and evaluate f(x) at the
        nodes

N = 10;
x = linspace(-5,5, N+1);
f = inline('1./(1+x.*x)','x');
y = f(x);
figure;
plot(x,y,'o');
title('N+1 = 11 equally spaced data points');
ylim([0,1.2]);
```



Now generate a lot of points ti at which to evalueate f, and the interpolants for plotting

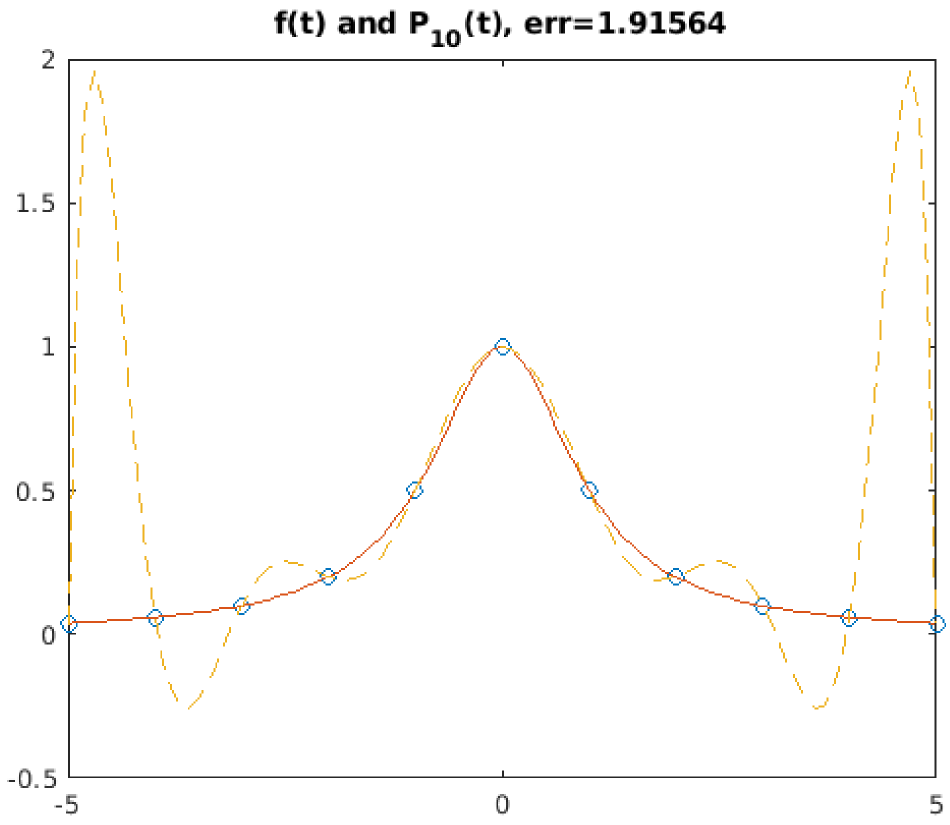
```
t = [-5:0.1:5];
figure;
hold on;
plot(t, f(t), '-')
plot(x,y,'o');
title('Runge function')
ylim([0,1.2]);
hold off;
```



2.0 Nth degree interpolating polynomial

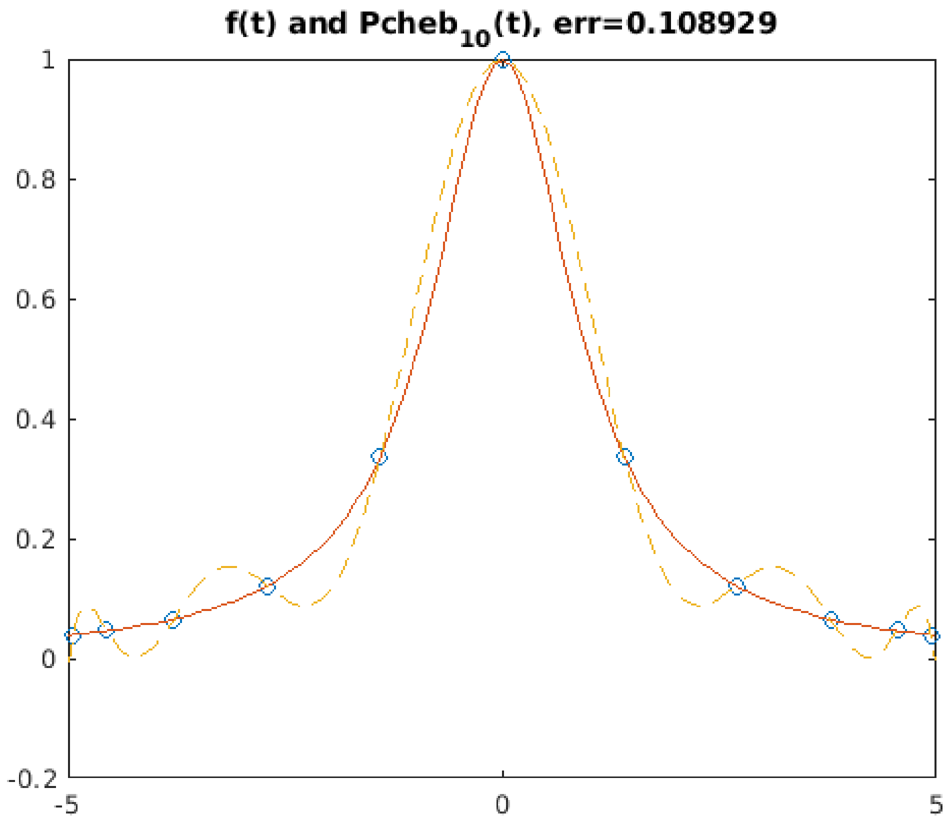
construct the Nth degree interpolating polynomial using the equally spaced nodes

```
PN = polyfit(x,y,N);
v = polyval(PN, t);
err = norm(f(t)-v, inf);
figure;
plot(x,y,'o', t, f(t), '-', t, v, '--');
title(sprintf('f(t) and P_{10}(t), err=%g', err));
```



3.0 Interpolation at Chebychev nodes:

```
        Generate N+1 = 11 Chebychev nodes

K = N + 1;
a=-5;
b= 5;
xcheb = zeros(1,K);
for i=1:K
    xcheb(i)=(a+b)/2 + (b-a)/2*cos((i-0.5)*pi/K);
end
ycheb = f(xcheb);
PNcheb = polyfit(xcheb, ycheb, N);
vcheb = polyval(PNcheb, t);
errcheb = norm(f(t)-vcheb, inf);

figure;
plot(xcheb,ycheb,'o', t, f(t), '-', t, vcheb, '--');
title(sprintf('f(t) and Pcheb_{10}(t), err=%g', errcheb));
```

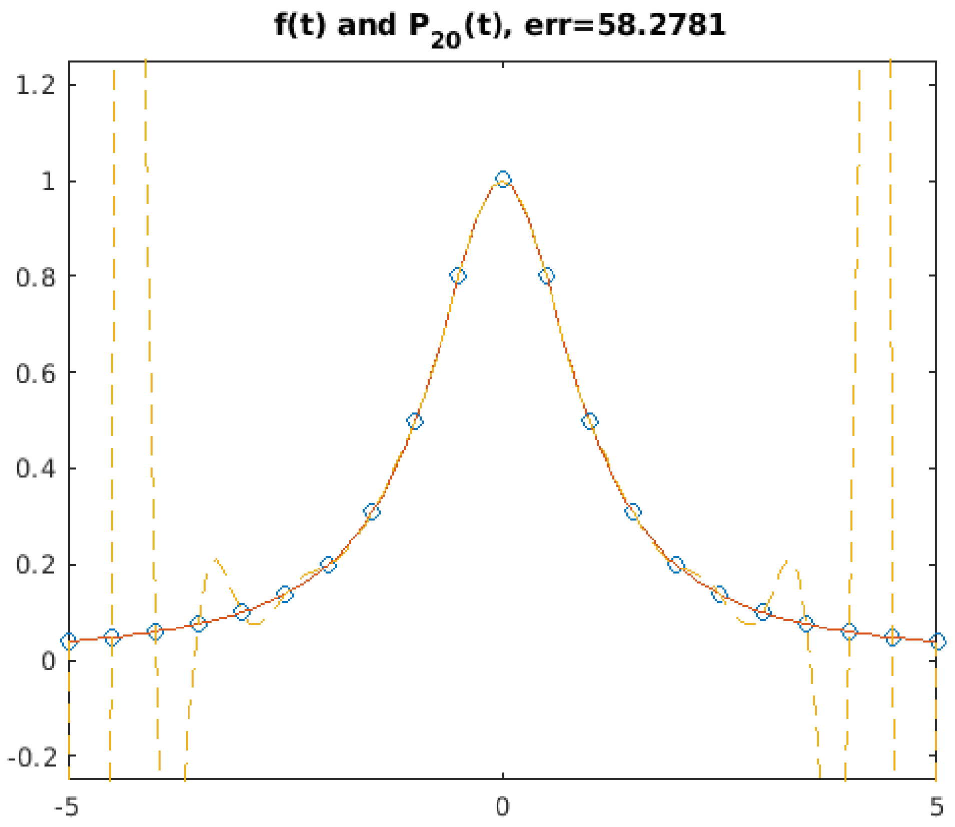


The error given by the Chebychev points was an order of magnitude less than the error given by the polyfit function for the Nth degree interpolating polynomial. Clearly the Chebychev interpolation worked better and this is because rather than choosing an evenly spaced grid, we cleverly choose an uneven grid according to the chebychev algorithm that works to minimize the error.

4.0 (a) -- Repeat with N=20

```
clear all;
N = 20;
x = linspace(-5,5, N+1);
f = inline('1./(1+x.*x)','x');
y = f(x);
t = [-5:0.1:5];

PN = polyfit(x,y,N);
v = polyval(PN, t);
err = norm(f(t)-v, inf);
figure;
plot(x,y,'o', t, f(t), '-', t, v, '--');
title(sprintf('f(t) and P_{20}(t), err=%g', err));
ylim([-0.25,1.25]);
```

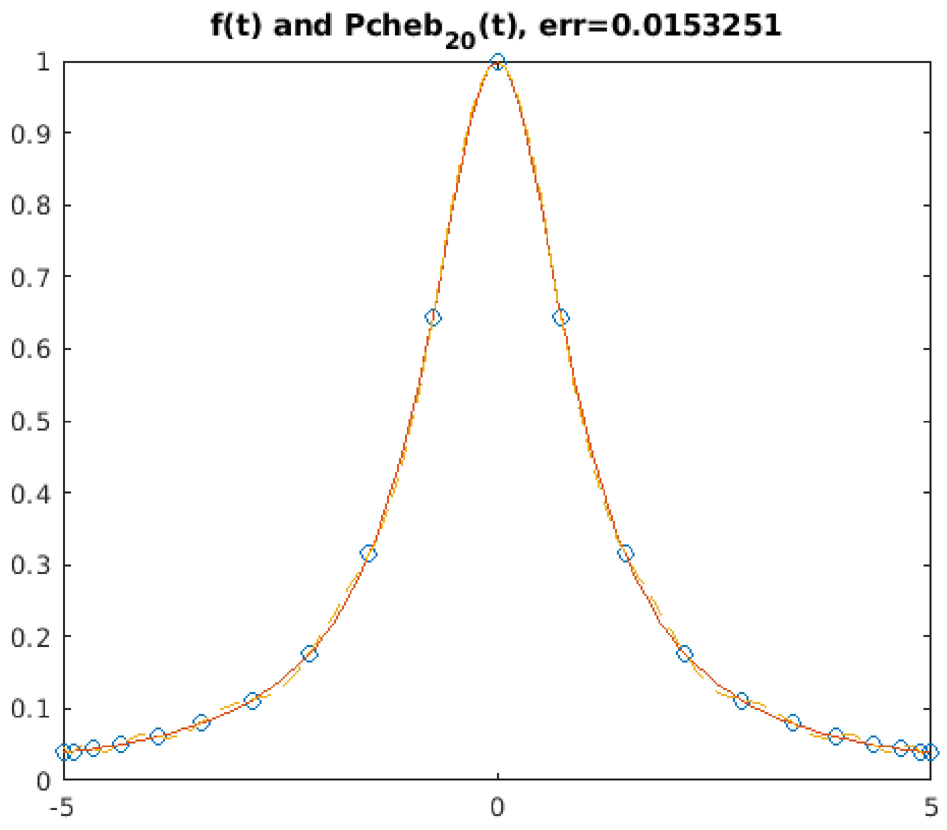
```
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
```



This appears to be a better fit towards the middle of the graph but the n th degree polynomial with even roots starts to blow up as we approach the edge of our interval. This resulted in the error increasing dramatically due to the approximation worsening towards the edge of the interval.

```
K = N+1;
a=-5;
b=5;
xcheb = zeros(1,K);

for i=1:K
    xcheb(i)=(a+b)/2 + (b-a)/2*cos((i-0.5)*pi/K);
end

ycheb = f(xcheb);
PNcheb = polyfit(xcheb, ycheb, N);
vcheb = polyval(PNcheb, t);
errcheb = norm(f(t)-vcheb, inf);

figure;
plot(xcheb,ycheb,'o', t, f(t), '-', t, vcheb, '--');
title(sprintf('f(t) and Pcheb_{20}(t), err=%g', errcheb));
```

```
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
```

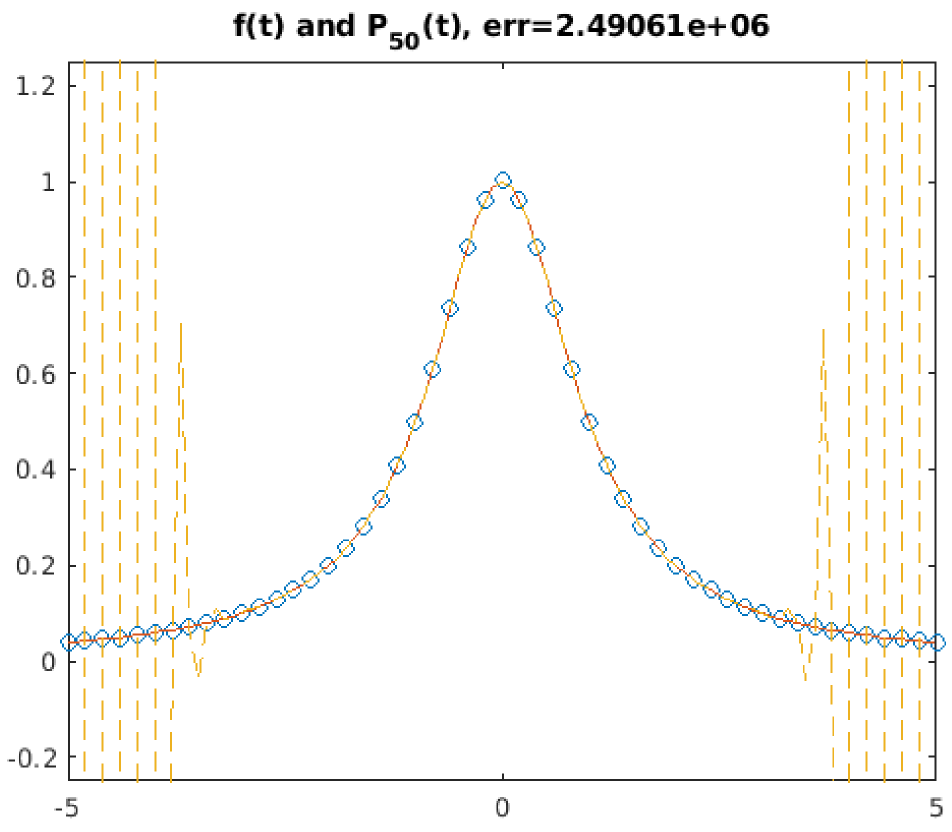


The Chebychev polynomial error decreased by an order of magnitude when we doubled N.

4.0 (b) -- Repeat with N = 50

```
clear all;
N = 50;
x = linspace(-5,5, N+1);
f = inline('1./(1+x.*x)','x');
y = f(x);
t = [-5:0.1:5];

PN = polyfit(x,y,N);
v = polyval(PN, t);
err = norm(f(t)-v, inf);
figure;
plot(x,y,'o', t, f(t), '-', t, v, '--');
title(sprintf('f(t) and P_{50}(t), err=%g', err));
ylim([-0.25,1.25]);
```

```
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
```

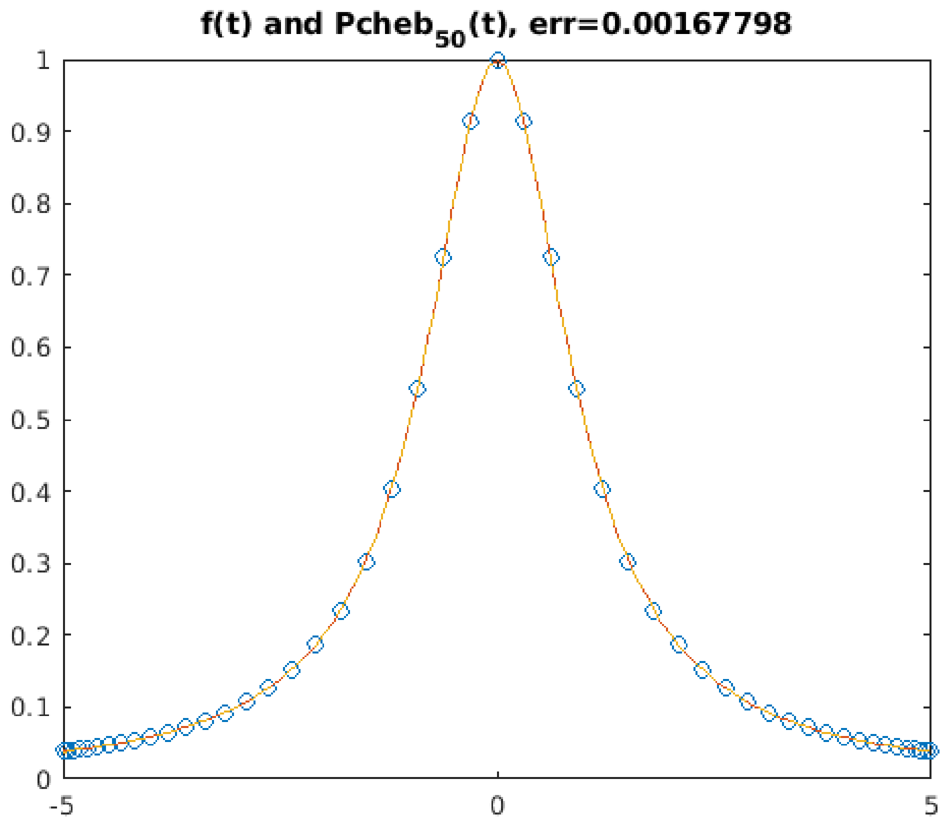


Now we can see that the polynomial interpolation is really bad towards the edge of the interval. It blows up really fast leading to an error of the order 10e6.

```
K = N+1;
a=-5;
b=5;
xcheb = zeros(1,K);

for i=1:K
    xcheb(i)=(a+b)/2 + (b-a)/2*cos((i-0.5)*pi/K);
end

ycheb = f(xcheb);
PNcheb = polyfit(xcheb, ycheb, N);
vcheb = polyval(PNcheb, t);
errcheb = norm(f(t)-vcheb, inf);

figure;
plot(xcheb,ycheb,'o', t, f(t), '-', t, vcheb, '--');
```

```
title(sprintf('f(t) and Pcheb_{50}(t), err=%g', errcheb));
```

```
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
```



The chebychev polynomial error decreased by another order of magnitude now that the number of points N = 50. It is extremely difficult to see the difference between the original function and the chebychev polynomial interpolation.

Published with MATLAB® R2017b