

# **MATH 446**

## **Project 5**

April 20, 2023

## Project Report

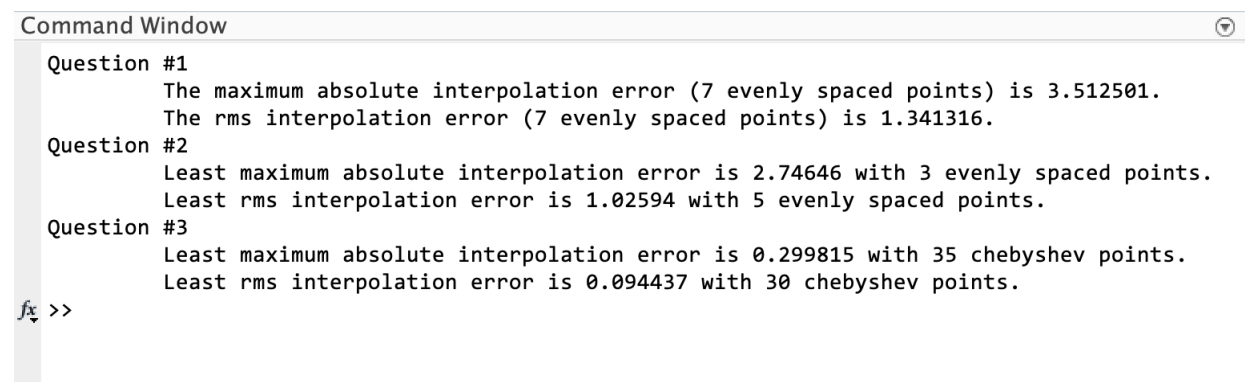
A function  $f(x) = \tan(\sin(x^3))$  is interpolated on the interval  $[0, 3]$  with evenly spaced and Chebyshev points. We have used the textbook functions `newtdd.m` and `nest.m` respectively to compute the coefficients of the interpolating polynomial and for the computation of the polynomial on a grid.

From the obtained results, we see that the errors are decreased when using Chebyshev points compared to evenly spaced points.

The least error when using evenly spaced points occurs with 3 points and the error is 2.74646. The errors are typically the largest near  $x = 0$ . The least error when using Chebyshev points occurs with 35 Chebyshev points, and the least error is 0.299815.

The improvement in quality of interpolation though is not expressed well by these numbers, since the errors are the maximum absolute error at some location in  $[0, 3]$ . The real benefit of using Chebyshev points over evenly spaced points is best shown by the plots of function vs the interpolant which are included below. We have also included the rms errors for better representation of interpolation quality over the entire interval.

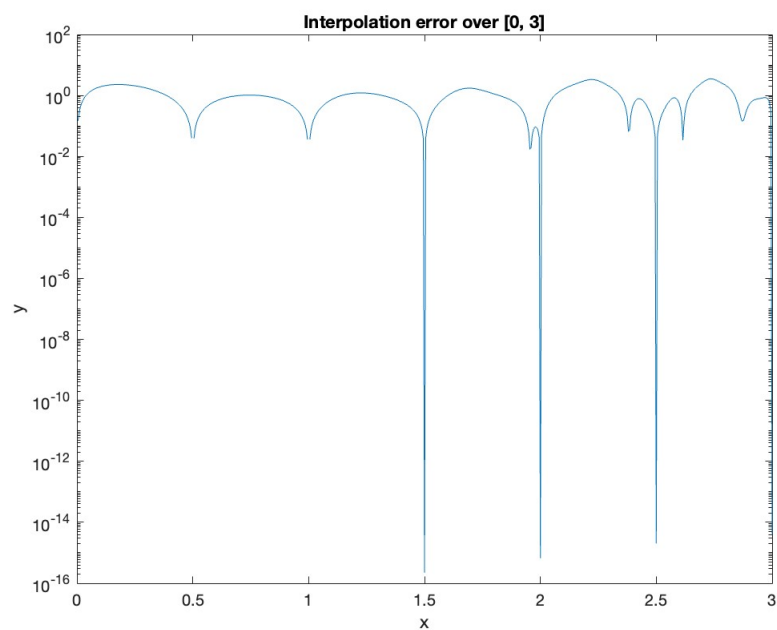
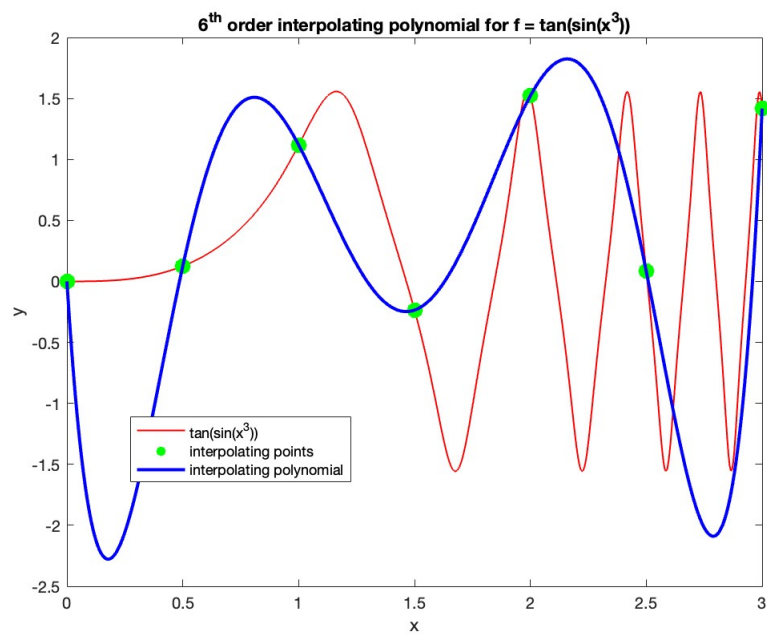
The following MATLAB Command Window Output summarizes the results.

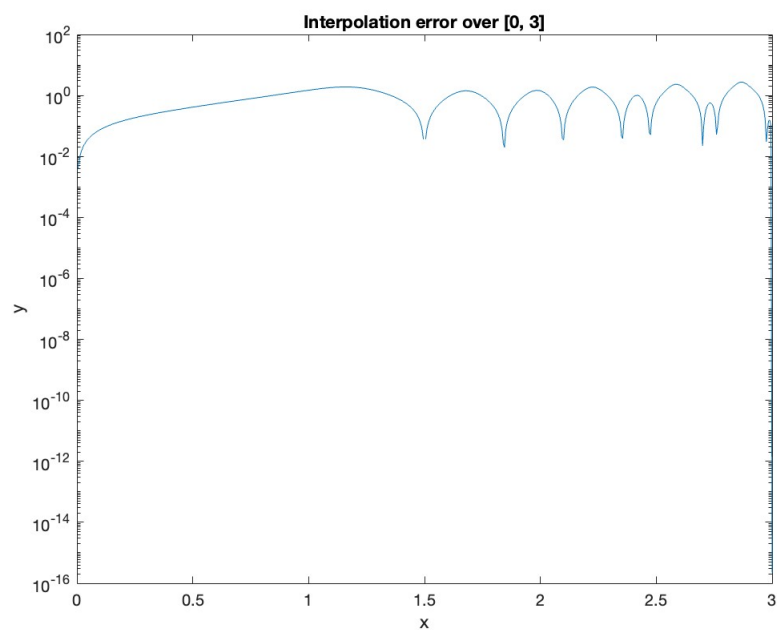
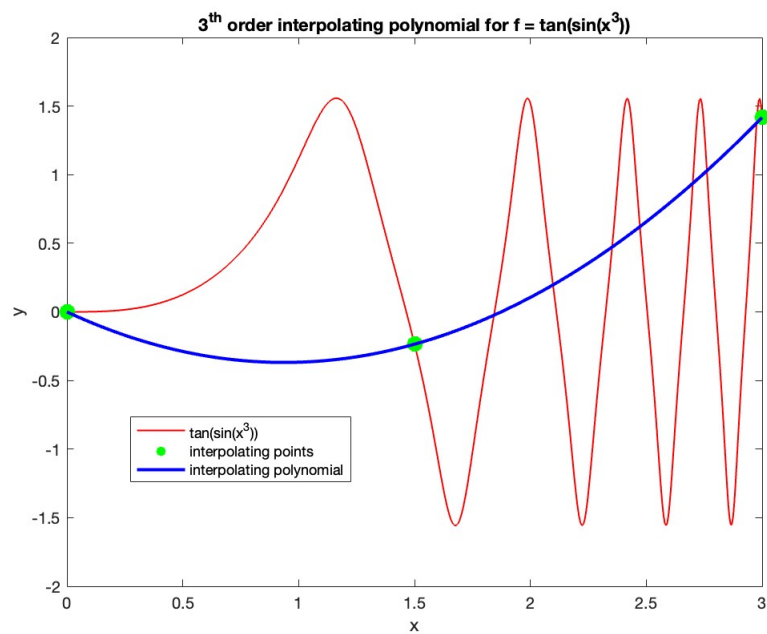


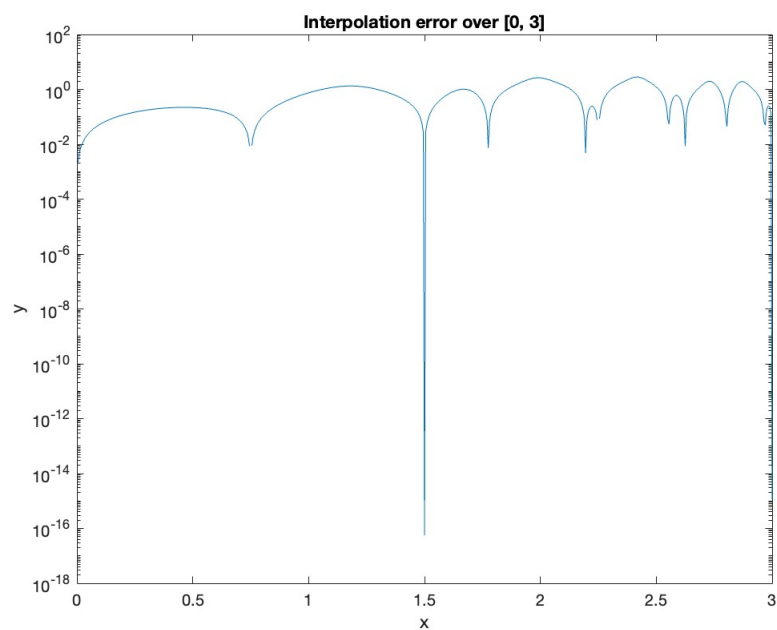
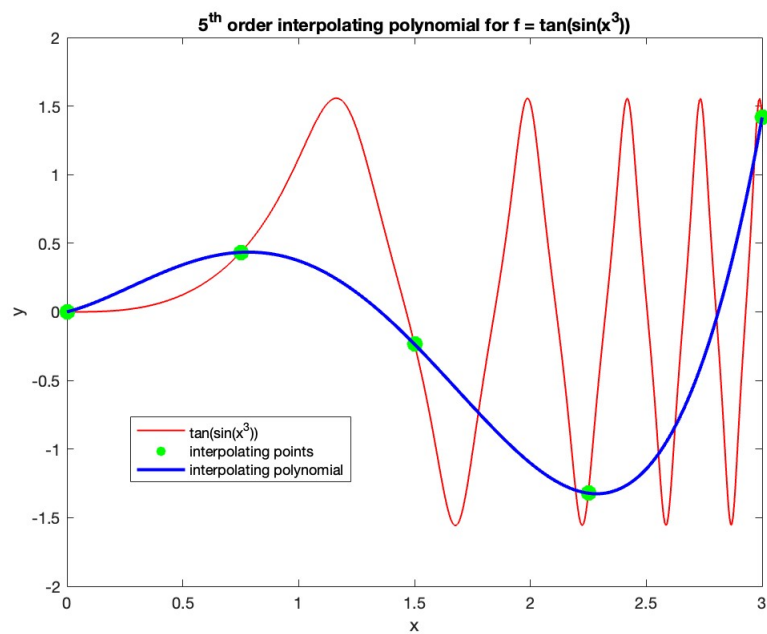
```
Command Window
Question #1
    The maximum absolute interpolation error (7 evenly spaced points) is 3.512501.
    The rms interpolation error (7 evenly spaced points) is 1.341316.
Question #2
    Least maximum absolute interpolation error is 2.74646 with 3 evenly spaced points.
    Least rms interpolation error is 1.02594 with 5 evenly spaced points.
Question #3
    Least maximum absolute interpolation error is 0.299815 with 35 chebyshev points.
    Least rms interpolation error is 0.094437 with 30 chebyshev points.
fx >>
```

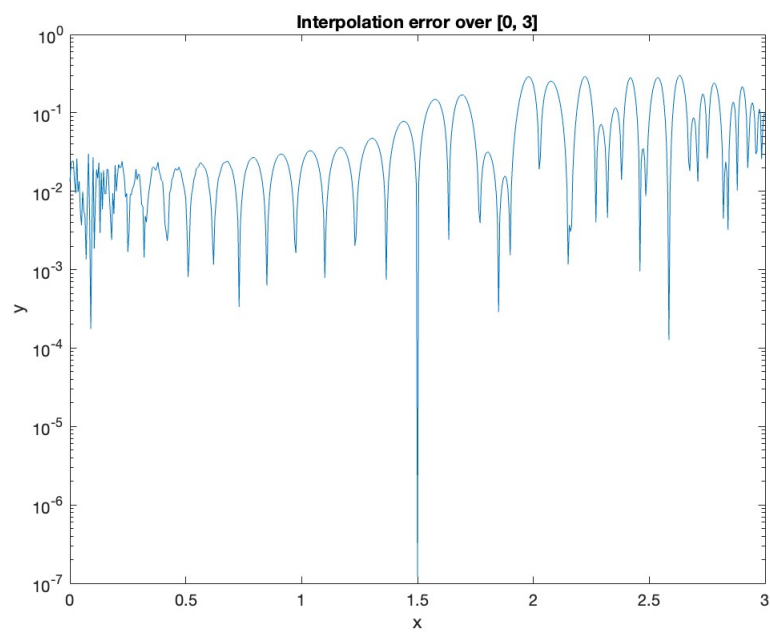
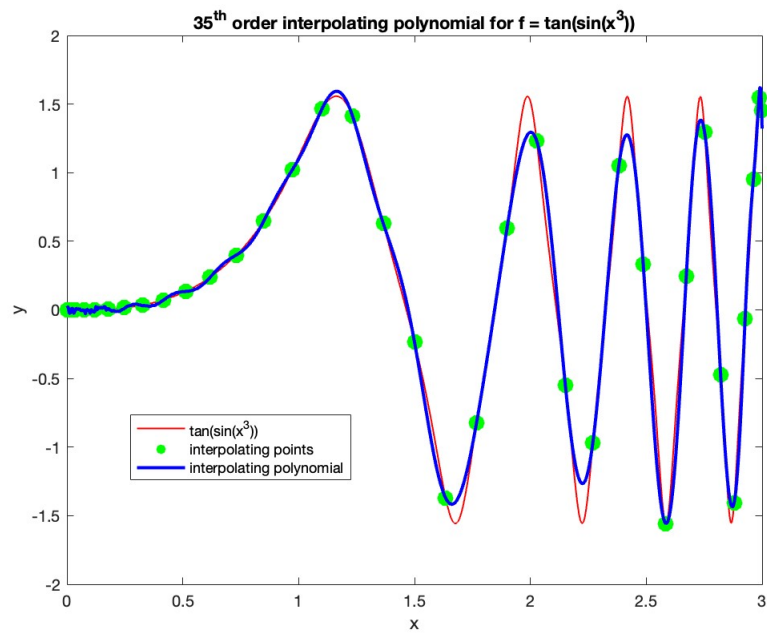
The errors get larger and larger as we increase the number of evenly spaced points, so much so, that the best error is given by a 3 point interpolant. This is due to Runge's phenomenon that happens at the extremes. The graph of  $f$  is quite flat near  $x = 0$  and the polynomial interpolation using evenly spaced points is unable to capture this feature, leading to large errors in this region.

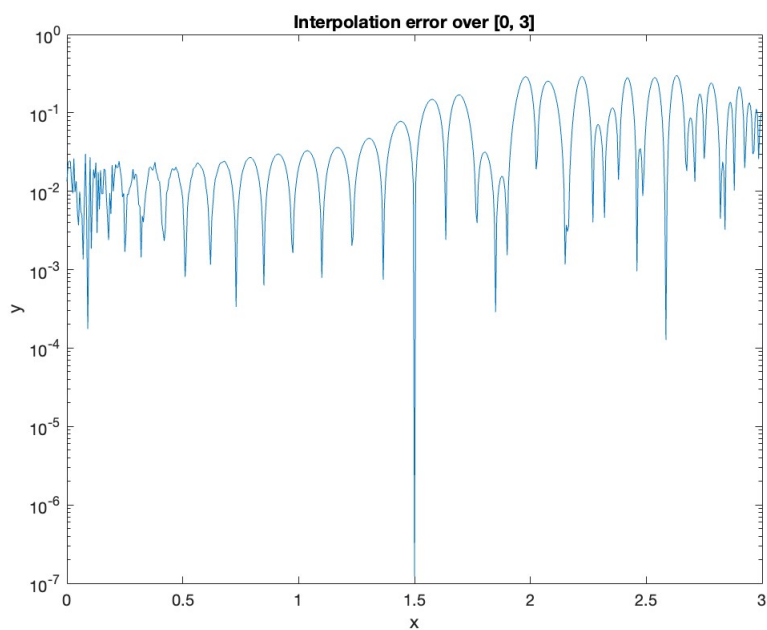
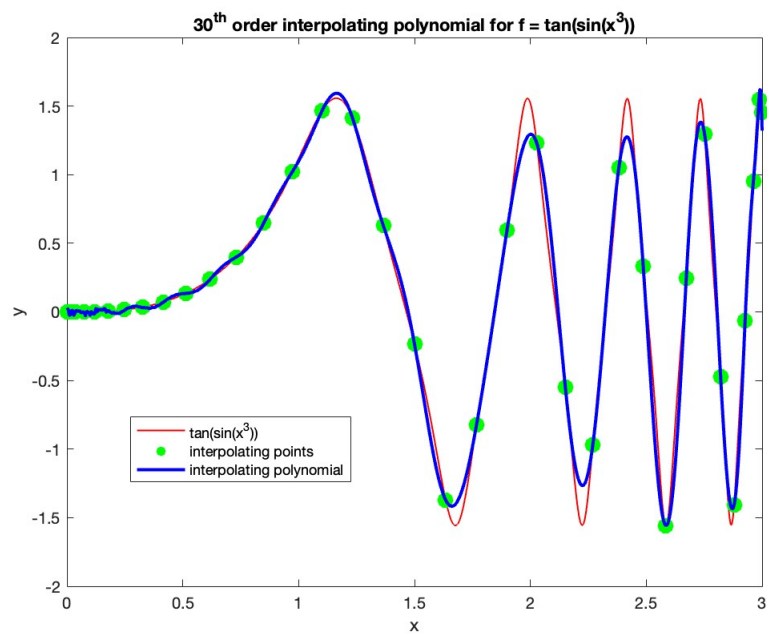
The interpolating polynomial plots and the interpolation errors plots follow.











## MATLAB Code - nest.m

```
% Program 0.1 Nested multiplication
% Evaluates polynomial from nested form using Horner s Method
% Input: degree d of polynomial,
%         array of d+1 coefficients c (constant term first),
%         x-coordinate x at which to evaluate, and
%         array of d base points b, if needed
% Output: value y of polynomial at x
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
```



## MATLAB Code - newtdd.m

```
% Program 3.1 Newton Divided Difference Interpolation Method
% Computes coefficients of interpolating polynomial
% Input: x and y are vectors containing the x and y coordinates
%       of the n data points
% Output: coefficients c of interpolating polynomial in nested form
% Use with nest.m to evaluate interpolating polynomial
function c = newtdd(x,y,n)
    for j=1:n
        v(j,1)=y(j); % Fill in y column of Newton triangle
    end

    for i=2:n % For column i,
        for j=1:n+1-i % fill in column from top to bottom
            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); % Read along top of triangle
    end % for output coefficients
end
```

## MATLAB Code - proj\_5.m

```
clc; clear; close all

%% interpolating with evenly spaced points in [0, 3]

n = 7;
x0 = 3*(0 : (n-1))/(n-1);
y0 = tan(sin(x0.^3));

% compute interpolating polynomial
c = newtdd(x0, y0, n);

% compute polynomial over a grid
x = 0:0.005:3;
p = nest(n-1, c, x, x0);
f = tan(sin(x.^3));

% plot function vs interpolant
figure()
plot(x, f, 'r-', 'LineWidth', 1)
hold on
scatter(x0, y0, 100, 'filled', 'green')
plot(x, p, 'b-', 'LineWidth', 2)
hold off
xlabel('x')
ylabel('y')
legend('tan(sin(x^3))', 'interpolating points', ...
'interpolating polynomial', 'location', 'best')
title('6th order interpolating polynomial for f = tan(sin(x^3))')

% plot interpolation error
interperror = abs(f - p);
figure()
semilogy(x, interperror)
xlabel('x')
ylabel('y')
title('Interpolation error over [0, 3]')

% compute maxerror
maxerror = max(interperror);
```

```

rmserror = rms(interperror);
fprintf(['Question #1 \n\t The maximum absolute interpolation error (7 evenly spaced'
' points) is %f.'], maxerror)
fprintf(['\n\t The rms interpolation error (7 evenly spaced' ...
' points) is %f.\n'], rmserror)

%% find 'n' that results in smallest possible interpolation error

% evenly spaced points
% take n = 30 as upper limit

maxerror = 1E16;
maxrmserror = 1E16;
nmax = 30;
nbest = nmax;
nrmsbest = nmax;
for n = nmax:-1:2
    x0 = 3 * (0 : (n-1))/(n-1);
    y0 = tan(sin(x0.^3));

    % compute interpolating polynomial
    c = newtdd(x0, y0, n);

    % compute polynomial over a grid
    x = 0:0.005:3;
    p = nest(n-1, c, x, x0);
    f = tan(sin(x.^3));

    % max interpolation error
    interperror = abs(f - p);
    if max(interperror) < maxerror
        maxerror = max(interperror);
        nbest = n;
    end
    if rms(interperror) < maxrmserror
        maxrmserror = rms(interperror);
        nrmsbest = n;
    end
end
end

```

```

fprintf(['Question #2 \n\t Least maximum absolute interpolation error is %g with %d e
' spaced points.'], maxerror, nbest)
fprintf(['\n\t Least rms interpolation error is %g with %d evenly' ...
' spaced points.\n'], maxrmserror, nrmsbest)

% plot function vs interpolant
% compute interpolating polynomial
x0 = 3 * (0 : (nbest-1))/(nbest-1);
y0 = tan(sin(x0.^3));
c = newtdd(x0, y0, nbest);

% compute polynomial over a grid
x = 0:0.005:3;
p = nest(nbest-1, c, x, x0);
f = tan(sin(x.^3));

figure()
plot(x, f, 'r-', 'LineWidth', 1)
hold on
scatter(x0, y0, 100, 'filled', 'green')
plot(x, p, 'b-', 'LineWidth', 2)
hold off
xlabel('x')
ylabel('y')
legend('tan(sin(x^3))', 'interpolating points', ...
'interpolating polynomial', 'location', 'best')
title([num2str(nbest), '^th order interpolating polynomial for f = tan(sin(x^3))'])

% plot interpolation error
interperror = abs(f - p);
figure()
semilogy(x, interperror)
xlabel('x')
ylabel('y')
title('Interpolation error over [0, 3]')

% plot function vs interpolant
% compute interpolating polynomial
x0 = 3 * (0 : (nrmsbest-1))/(nrmsbest-1);
y0 = tan(sin(x0.^3));

```

```

c = newtdd(x0, y0, nrmsbest);

% compute polynomial over a grid
x = 0:0.005:3;
p = nest(nrmsbest-1, c, x, x0);
f = tan(sin(x.^3));

figure()
plot(x, f, 'r-', 'LineWidth', 1)
hold on
scatter(x0, y0, 100, 'filled', 'green')
plot(x, p, 'b-', 'LineWidth', 2)
hold off
xlabel('x')
ylabel('y')
legend('tan(sin(x^3))', 'interpolating points', ...
'interpolating polynomial', 'location', 'best')
title([num2str(nrmsbest), '^th order interpolating polynomial for f = tan(sin(x^3))

% plot interpolation error
interperror = abs(f - p);
figure()
semilogy(x, interperror)
xlabel('x')
ylabel('y')
title('Interpolation error over [0, 3]')

%%
% find 'n' that results in smallest possible interpolation error

% using chebyshev points
% take n = 40 as upper limit

maxerror = 1E16;
maxrmerror = 1E16;
nmax = 40;
nbest = nmax;
nrmsbest = nmax;
for n = nmax:-1:2
x0 = 3/2 + 3/2*cos((1:2:2*n-1)*pi/(2*n));
y0 = tan(sin(x0.^3));

```

```

% compute interpolating polynomial
c = newtdc(x0, y0, n);

% compute polynomial over a grid
x = 0:0.005:3;
p = nest(n-1, c, x, x0);
f = tan(sin(x.^3));

% max interpolation error
interperror = abs(f - p);
if max(interperror) < maxerror
    maxerror = max(interperror);
    nbest = n;
end
if rms(interperror) < maxrmserror
    maxrmserror = rms(interperror);
    nrmsbest = n;
end
end

fprintf(['Question #3 \n\t Least maximum absolute interpolation error is %g with %d ',
'chebyshev points.'], maxerror, nbest)
fprintf(['\n\t Least rms interpolation error is %g with %d ' ...
'chebyshev points.\n'], maxrmserror, nrmsbest)

% plot function vs interpolant
% compute interpolating polynomial
x0 = 3/2 + 3/2*cos((1:2:2*nbest-1)*pi/(2*nbest));
y0 = tan(sin(x0.^3));
c = newtdc(x0, y0, nbest);

% compute polynomial over a grid
x = 0:0.005:3;
p = nest(nbest-1, c, x, x0);
f = tan(sin(x.^3));

figure()
plot(x, f, 'r-', 'LineWidth', 1)
hold on
scatter(x0, y0, 100, 'filled', 'green')

```

```

plot(x, p, 'b-', 'LineWidth', 2)
hold off
xlabel('x')
ylabel('y')
legend('tan(sin(x^3))', 'interpolating points', ...
'interpolating polynomial', 'location', 'best')
title([num2str(nbest), '^{th} order interpolating polynomial for f = tan(sin(x^3))'])

% plot interpolation error
interperror = abs(f - p);
figure()
semilogy(x, interperror)
xlabel('x')
ylabel('y')
title('Interpolation error over [0, 3]')

% plot function vs interpolant
% compute interpolating polynomial
x0 = 3/2 + 3/2*cos((1:2:2*nbest-1)*pi/(2*nbest));
y0 = tan(sin(x0.^3));
c = newtdd(x0, y0, nbest);

% compute polynomial over a grid
x = 0:0.005:3;
p = nest(nbest-1, c, x, x0);
f = tan(sin(x.^3));

figure()
plot(x, f, 'r-', 'LineWidth', 1)
hold on
scatter(x0, y0, 100, 'filled', 'green')
plot(x, p, 'b-', 'LineWidth', 2)
hold off
xlabel('x')
ylabel('y')
legend('tan(sin(x^3))', 'interpolating points', ...
'interpolating polynomial', 'location', 'best')
title([num2str(nrmsbest), '^{th} order interpolating polynomial for f = tan(sin(x^3))'])

% plot interpolation error
interperror = abs(f - p);

```

```
figure()
semilogy(x, interperror)
xlabel('x')
ylabel('y')
title('Interpolation error over [0, 3]')
```