Ngoc Huynh
MATH 446 / OR 481
Project 5
Approximating functions by polynomial interpolation

$$f(x) = sin7x + cos4x$$

Question 1:

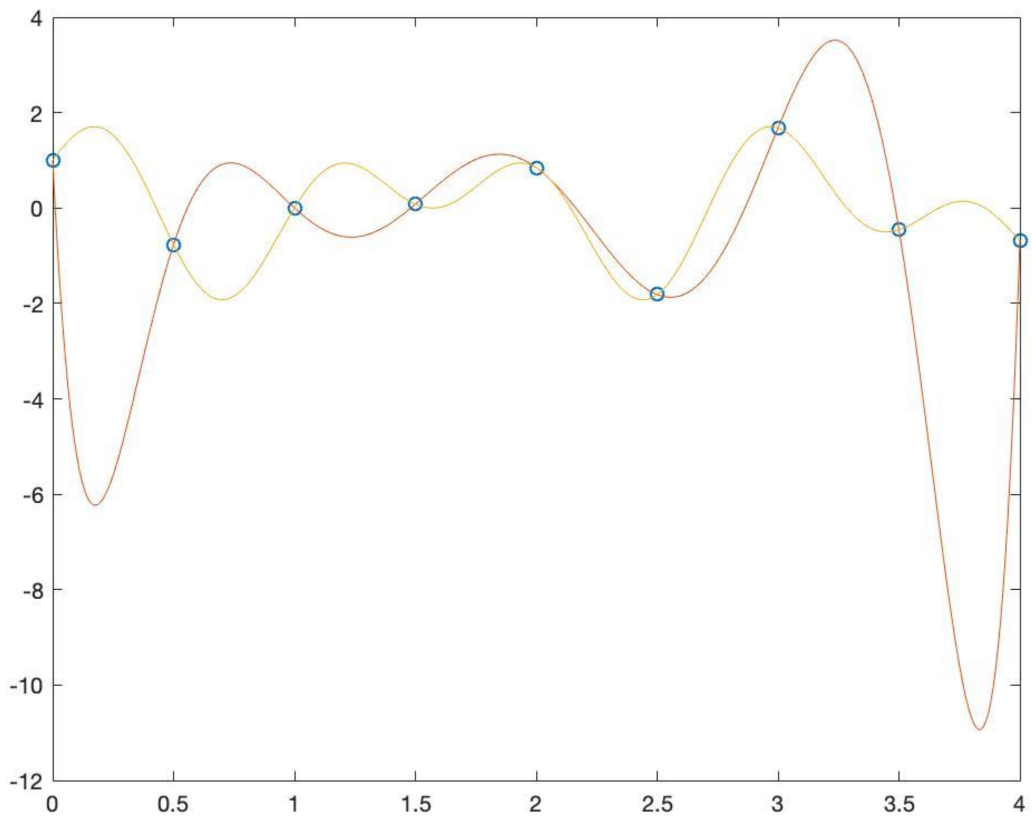
Matlab Code:

```
%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);
end
for i=2:n
  for j=1:n+1-i
% Fill in y column of Newton triangle
% For column 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
```

```
%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
```

```
% Ngoc Huynh
% Matlab Code - Project 5
f = @(x) sin(7*x) + cos(4*x);
n=9;
x0=4*(0:(n-1))/(n-1);
y0=f(x0);
c=newtdd(x0,y0,n);
x=0:.01:4;
y=nest(n-1,c,x,x0);
plot(x0,y0,'o',x,y,x,f(x))
maxerr=max(abs(y-f(x)));
```

Plot the actual f(x) versus P_{n-1}(x) on [0,4] for n = 9 points:

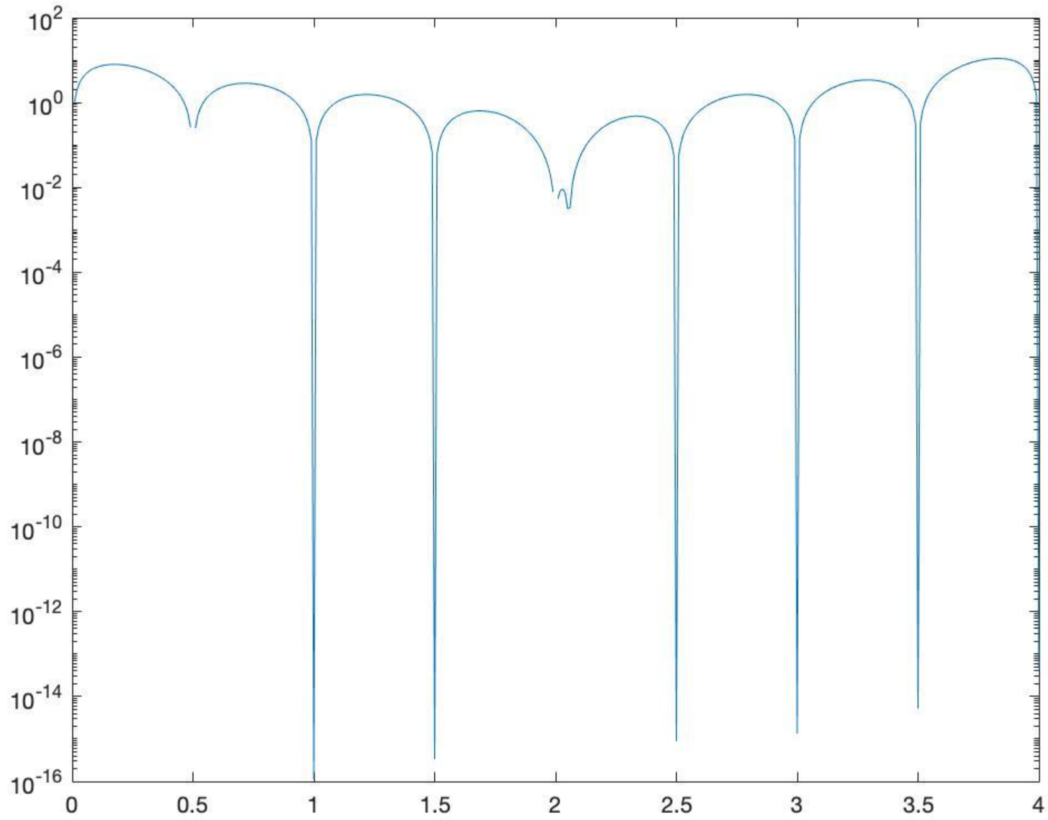


Calculate the maximum interpolation error on the domain [0,4], using MatLab:

```
maxerr =

   11.0027
```

Plot the interpolation error of P_8(x) on [0, 4], using Matlab's semiology command:

```
>> semilogy(x,abs(y-f(x)))
```



Question 2:

Find the smallest n that makes the maximum interpolation error on the domain [0, 4] less than 0.5×10^{-6}, which equals to 5×10^{-7}.

With n = 34, maximum error is:

```
maxerr =

    3.2839e-06
```

Which is still greater than 0.5×10^{-6}.

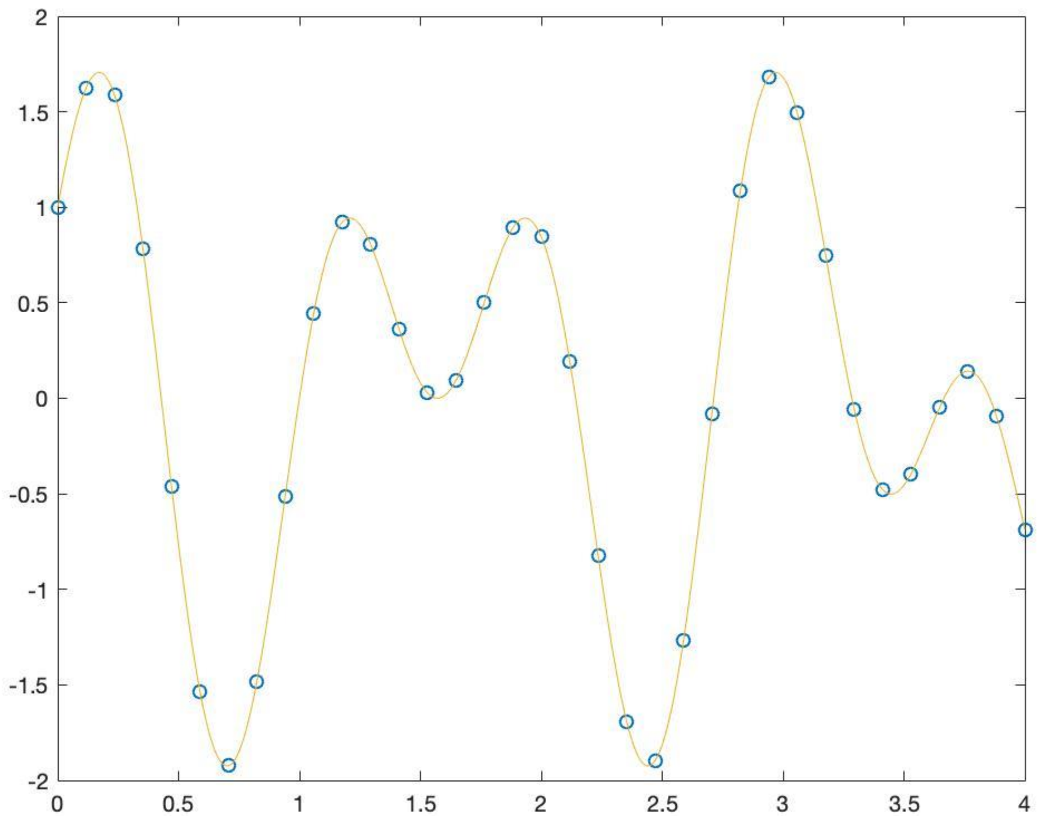
Now I tried **n = 35**, maximum error is:

```
maxerr =

    4.9799e-07
```

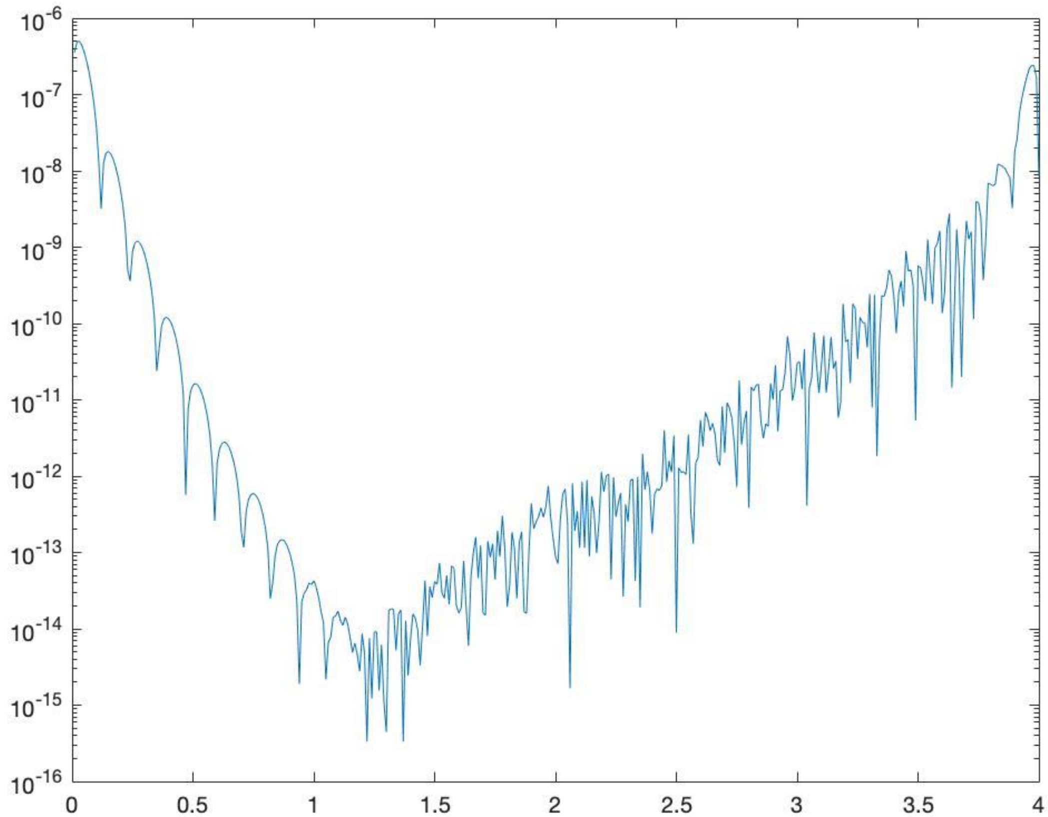
Which is precisely smaller than 0.5×10^{-6}.

→ smallest possible n is 35 because with any n < 35, the maximum errors returned are all greater than 0.5×10^{-6}

Interpolation polynomial plot:



The semilog interpolation error plot:



Along the interval [0, 4], the largest error typically is between 0 and 0.1.

<u>Question 3:</u>

Matlab Code:

```
% Ngoc Huynh
% Matlab Code - Project 5
f = @(x) sin(7*x) + cos(4*x);
n=29;
x0=2+2*cos((1:2:2*n-1)*pi/(2*n));
y0=f(x0);
c=newtdd(x0,y0,n);
x=0:.01:4;
y=nest(n-1,c,x,x0);
plot(x0,y0,'o',x,y,x,f(x))
maxerr=max(abs(y-f(x)));
```

Find the smallest n that makes the maximum interpolation error on the domain [0, 4] less than 0.5 x 10^{-6}, which equals to 5 x 10^{-7}.
With **n = 28**, maximum error is:

```
maxerr =

    5.9631e-07
```

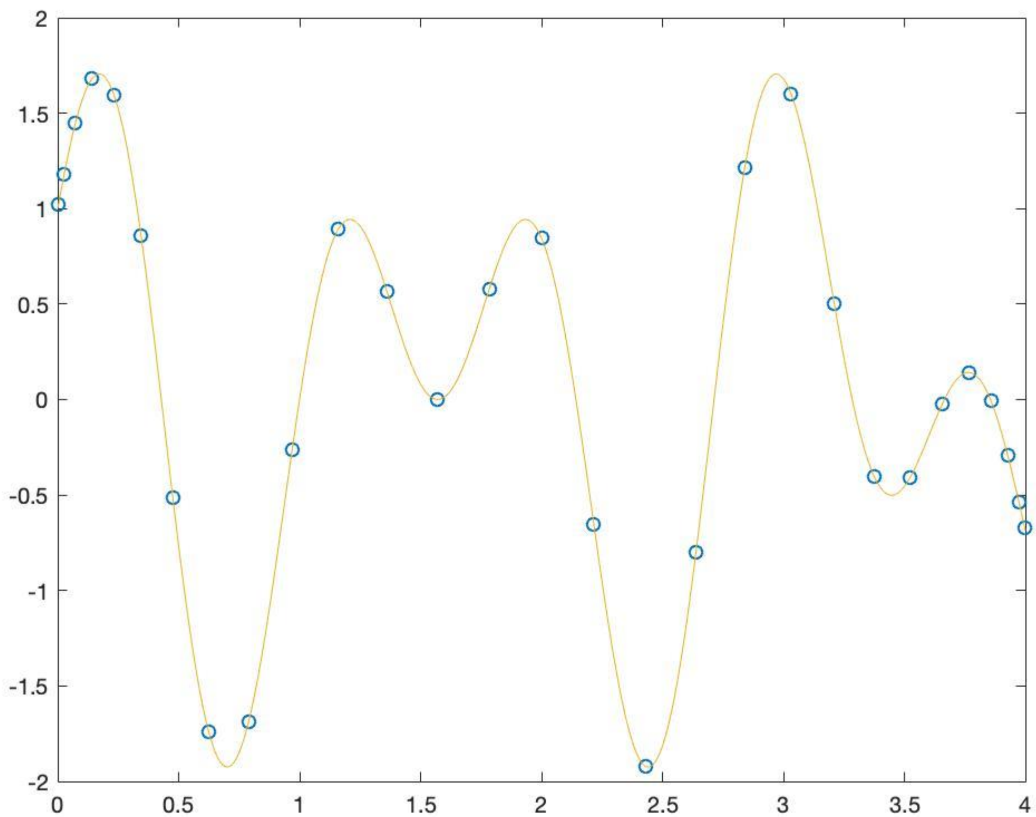
Which is still greater than 0.5 x 10^{-6}.

Now I tried **n = 29**, maximum error is:

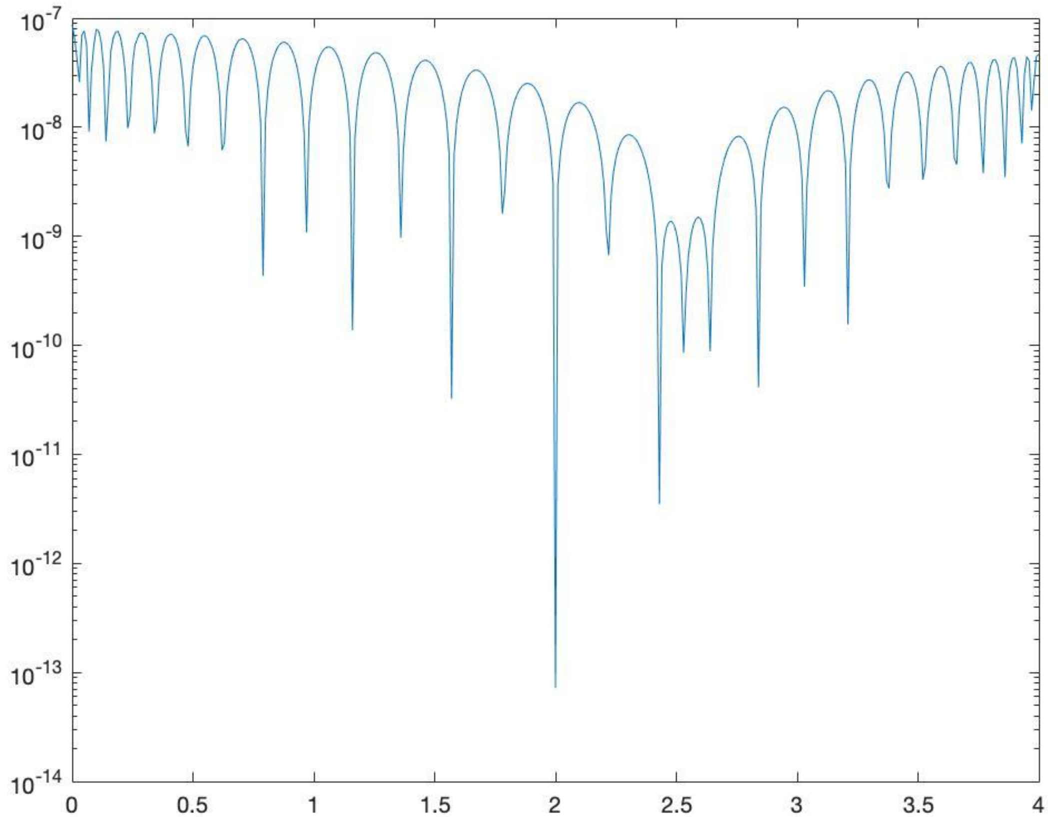
```
maxerr =

    8.7152e-08
```

→ **smallest n is 29**. because with any n < 29, the maximum errors are all greater than 0.5 x 10^{-6}.

Interpolation polynomial plot:



The semilog interpolation error plot:



Along the interval [0, 4], the largest error typically is between 0 and 0.4, becoming largest when getting closer to 0.

Question 4:

Comparing between interpolation with equally-spaced interpolation points and Chebyshev points → Chebyshev points was much better than equally-spaced interpolation points.
Because with equally-spaced interpolation points, I notice a large oscillations near the end of the interval, which indicates the Runge's phenomenon. With the same set of function, using Chebyshev points, I can see that the oscillations near the end of interval have been greatly reduced.